HTL | MÖSSINGERSTRASSE

HÖHERE TECHNISCHE BUNDESLEHRANSTALT

KLAGENFURT, MÖSSINGERSTRASSE

Abteilung ELEKTRONIK

Ausbildungsschwerpunkt Technische Informatik

INGENIEURPROJEKT

Webserver für LG500

Webserver for LG500

bearbeitet von Robert Stocker

Projektbetreuer Prof. Dipl.-Ing. Robert Oyrer

Prof. Dipl.-Ing. Dr. Günther Platzer

Inhaltsverzeichnis

1.1 Projektergebnis

1.1.1 Realisierte Funktionen:

- Ein- und Ausschalten des LG 500
- Auf Lüfterstufe 4 schalten
- Auf Lüfterstufe 1 schalten
- Soll-Wert für Temperatur einstellen
- Auf Automatikbetrieb umschalten
- Anzeige der aktuellen Ist- und Soll-Temperaturen
- Filterrestlaufzeit anzeigen

1.1.2 Resümee der gesamten Umsetzung

Die Vision, jene Informationen, die der Benutzer auf dem Bedienteil ablesen kann auch übers Internet oder Intranet über eine Webseite sehen zu können, konnte, wie beabsichtigt, zu 100% umgesetzt werden.

Die Funktionen des manuellen Stoßlüftens (Lüfterstufe IV), das Aus- und Einschalten der Anlage sowie das Einschalten des Automatikbetriebes der Anlage wurde ebenfalls zu 100% umgesetzt.

Dabei mussten an der bestehenden Steuerung des LG 500 weder an der Hardware, noch an der Software Änderungen vorgenommen werden.

Da das LG500 ohnehin eine bedarfsgeführte mechanische Be- und Entlüftungsanlage ist, die sowohl CO2-Werte als auch Temperaturen und Tageszeiten berücksichtigt, wurden diverse sinnlose Funktionen, die zu Anfang ohne eingehende Prüfung in das Pflichtenheft geschrieben wurden, revidiert und aus dem Pflichtenheft genommen.

Somit konnte das Projekt Miniwebserver für das LG500 zu 100% erfolgreich abgeschlossen werden.

Schließlich wurde der Prototyp des Miniwebservers noch in ein Gehäuse eingebaut.

2 Vorprojektphase

2.1 Projektvision

Der Benutzer einer Hauslüftung soll von überall auf der Welt die aktuellen Daten seiner Hauslüftung über das Internet abrufen und die Parameter verändern können. Das macht das Lüftungssystem interessanter und praktischer für Menschen, die nicht immer vor Ort sind.

2.2 Industriekontakte und Praxisbezug

Das Projekt wird für die Firma Pichler Lufttechnik entwickelt und soll in der Praxis Einsatz finden.

2.3 Projektumfeld

In diesem Kapitel wird das Umfeld des Projektes beschrieben.

Das Projekt wurde von Herrn Ing. Grassler, dem technischen Leiter bei der Firma Pichler Lufttechnik in Auftrag gegeben. Mit Herrn Ing. Grassler wurden die Grundzüge des Projektes besprochen.

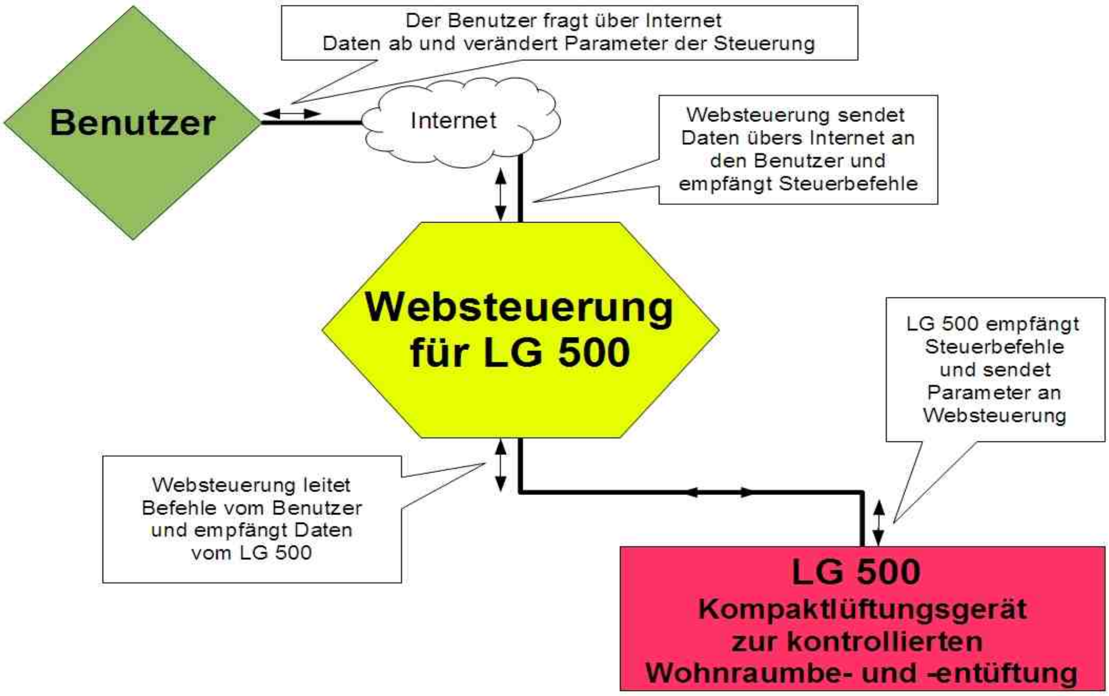
Von Herrn Ing. Grassler wurde ich an Herrn Pölzelbauer, der an der Steuerung arbeitet und das LG500 mitentwickelt verwiesen, der mir mit Rat und Tat zur Seite steht, mit dem die Einzelheiten besprochen wurden und der mich mit Informationen versorgt. Zusammen mit Herrn Pölzelbauer haben wir erarbeitet, welche Funktionen die Websteuerung haben sollte. Außerdem arbeitet Herr Pölzelbauer mit einem Modell, mit dem die Funktionen des Lüftungsgerätes simuliert werden können und die Steuerung somit getestet werden kann. Zu diesem Modell habe ich durch Herrn Pölzelbauer auch Zugang.

Des Weiteren habe ich Kontakt mit Herrn Senicar von Hermes Electronics in Deutschland bekommen, der die Steuerung programmiert und über technische Details sehr genau Bescheid weiß. Herr Senicar hat mir zugesagt, mich mit allen nötigen Informationen zu versorgen, hat schon mögliche Wege vorgezeichnet und ist bereit mir zu helfen und nötigenfalls auch Änderungen an der Steuersoftware vorzunehmen.

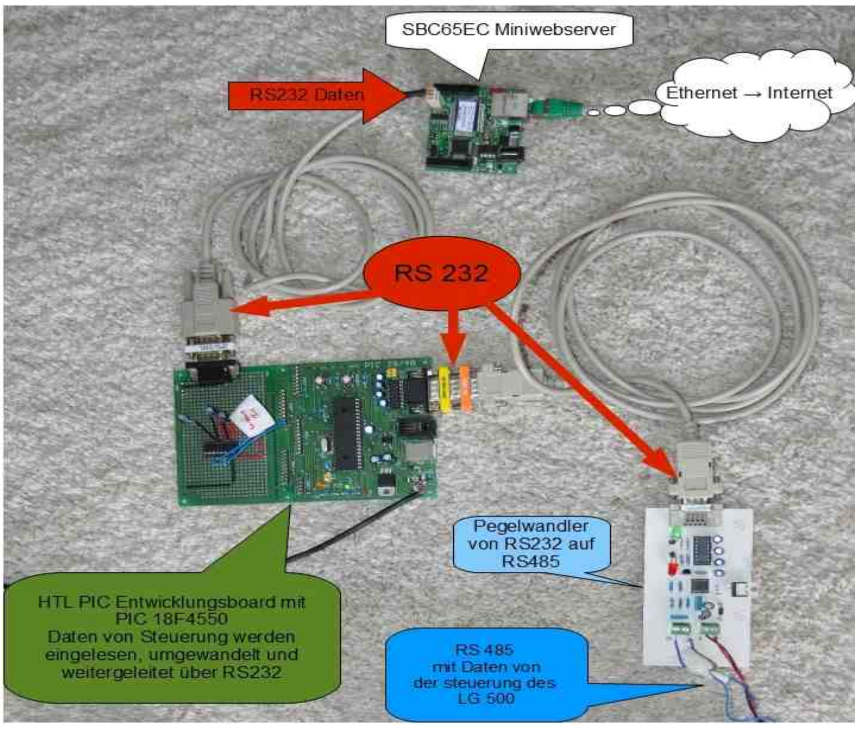
Die Websteuerung ist zwar ein selbständiges Projekt, hat aber möglicherweise auch kleinere Auswirkungen auf die Entwicklung des LG500.

2.4 Systemarchitektur

Schematisch:



Hardware:



2.5 Risikoabschätzung

Auflistung der Risiken in tabellarischer Form:

Name	**Wahr-schein-lichkeit**	**Auswirkungs-grad**	**Abhilfe**	**Kurzbeschreibung**
Verbindung SBC65EC mit LG 500	80,00%	hoch	Hilfestellung Fa. Pichler, bzw. Ventech und Fa. Hermes Electronic	Am Anfang des Projektes war nicht klar, ob eine Datenverbindung mit dem UZA Steuergerät des LG500 überhaupt möglich sein würde.
Programmierung für Verbindung	70,00%	hoch	Hilfestellung durch Betreuungslehrer	Die Programmierung der Software für eine Datenverbindung und das Einlesen und Extrahieren der benötigten Daten.
Verbindung SBC65EC mit Internet	5,00%	gering	Hilfestellung durch Betreuungslehrer	Es war nicht ganz klar, wie man den SBC65EC mit dem Internet verbinden kann.
Programmierung der Homepage	50,00%	mittel	Hilfestellung durch Betreuungslehrer bzw. Online Forum	Da die Homepage und die Serversoftware des SBC65EC mit dem C18 Compiler programmiert ist, bestand das Risiko, Schwierigkeiten beim Adaptieren der Software zu haben.

3 Pflichtenheft

Anzeige der für den Benutzer relevanten Daten der Lüftungsanlage auf der Homepage eines Webservers. Steuerung des LG 500 (Kompaktlüftungsgerät für kontrollierte Wohnraumbe- und -entlüftung) über das Internet ermöglichen.

● Funktionen:

- Ein- und Ausschalten
- Betriebsstufe 4 einstellen
- Soll-Wert für Temperatur einstellen
- Anzeige der aktuellen Ist- und Soll-Temperaturen
- Filterrestlaufzeit anzeigen

3.1 Funktionale und nicht funktionale Anforderungen

3.1.1 Funktionale Anforderungen

Der Miniwebserver fährt innerhalb einer Minute nach erfolgter Spannungsversorgung vollständig hoch, zeigt seine ordnungsgemäße Funktion durch Blinken einer roten LED an, stellt selbständig die Webseite über Ethernet zur Verfügung, fragt vor jedem Zugriff auf die Webseite den Benutzernamen und das Passwort ab, lässt sich durch Anklicken des Buttons „Update" aktualisieren und zeigt dann die Parameter an, die über die Datenleitung zwischen Bedienteil und LG500 bereits erfasst wurden. Weiters kann der Benutzer die in der UZA vordefinierten Funktionen über die Webseite schalten und die Lüftungsanlage reagiert innerhalb 20 Sekunden auf geschaltete Funktionen.

3.1.2 Nicht funktionale Anforderungen

Der Miniwebserver muss in ein bestehendes Datennetz mittels Ethernet-Verbindung eingebunden werden und muss in diesem Datennetz eine fixe IP-Adresse zugewiesen bekommen.

Der Miniwebserver ist von außen über das Internet nur dann zu erreichen, wenn der Benutzer eine bestehende Homepage mit fest zugewiesener IP-Adresse hat und den Miniwebserver mit dieser Homepage verlinkt, sodass die im Webserver fix eingestellte IP-Adresse für den Aufruf der Seite herangezogen werden kann.

3.1.3 Schnittstellen

Der Webserver benötigt eine Spannungsversorgung, wie sie üblicherweise von österreichischen Stromversorgungsunternehmen zur Verfügung gestellt wird. (230V, 50Hz)

Der Webserver wird über ein handelsübliches CAT5-Datenkabel mit der Steuerung des LG500 verbunden. Dabei werden 8 Leitungen für das Senden von Steuerbefehlen an das LG500 verwendet und 4 Leitungen für den Datenverkehr (RS485). Die Länge der Kabel sollte maximal ca. 100 m (laut Angabe von Hermes Electronic) betragen, d.h. die Lüftungsanlage sollte vom Miniwebserver nicht weiter weg platziert werden, um einen optimalen Empfang der Daten zu gewährleisten.

3.2 Abnahmekriterien

Für eine Prüfung der Funktionalität des Miniwebservers wird dieser, wie in der Dokumentation angegeben, an das LG500 angeschlossen und mit Spannung versorgt. Dann wird der Miniwebserver über ein Crossover-Ethernetkabel mit einem PC oder Laptop verbunden. Durch Aufruf der IP-Adresse des Miniwebservers (Default = 10.1.0.1) im Browser wird die Seite aufgerufen.

Durch Anklicken der Update-Funktion, nachdem die anzuzeigenden Daten von der UZA zur Verfügung gestellt wurden, werden die Daten der Lüftungsanlage auf der Homepage dargestellt. Durch Anklicken der Buttons in der Zeile „Funktionen“ werden die vordefinierten Funktionen der UZA geschalten.

Wenn all das funktioniert, hat der Miniwebserver den Systemtest bestanden.

3.3 Dokumentationsanforderungen

Keine besonderen Dokumentationsanforderungen.

3.4 Qualitätsstandards

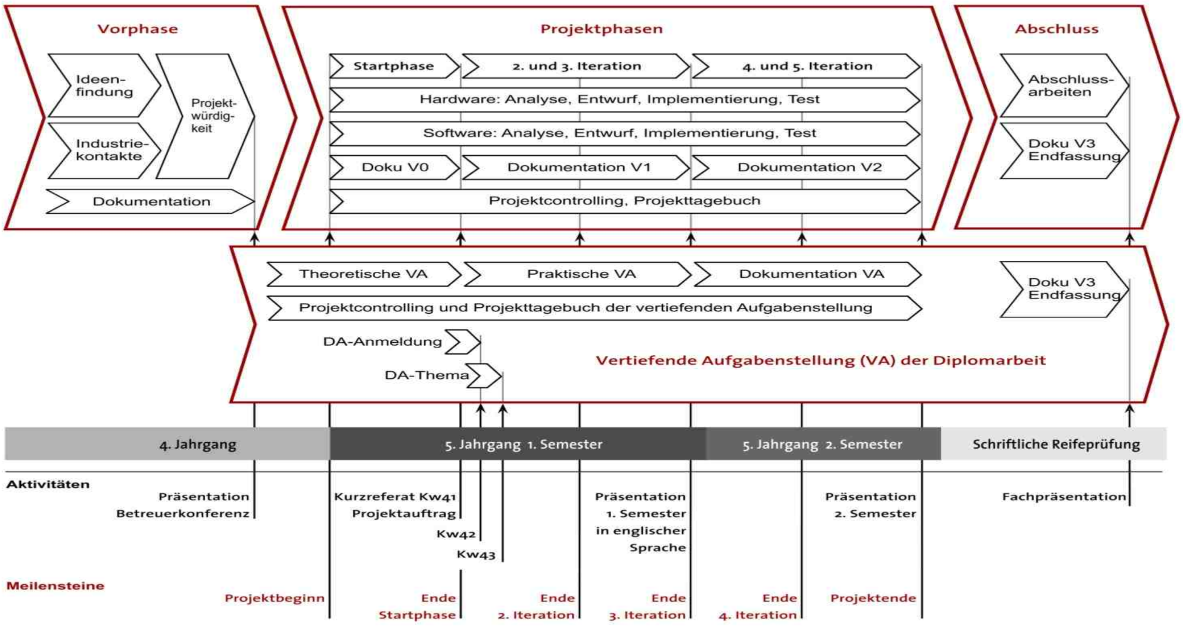
Da nur zertifizierte und zugelassene Bauteile und Microcontrollerplatinen verwendet werden, werden sämtliche Anforderungen und Normen der Stromnetzbetreiber eingehalten.

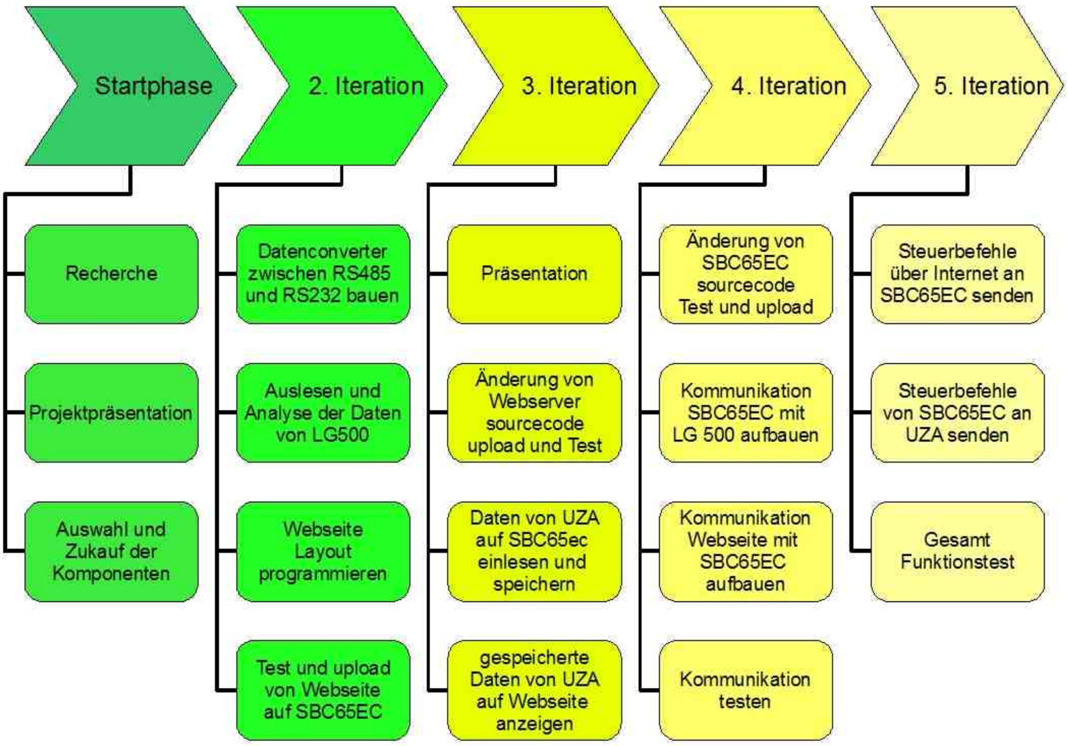
Die Einhaltung der EIA232-Standards für die webserverinterne Datenübertragung mit RS232 wird ebenfalls durch zertifizierte Bauteile sichergestellt.

Der Miniwebserver SBC65EC ist **ROHS konform**

3.5 Prozessmodell

Für die Projektabwicklung ist das an der HTL-Mössingerstraße Klagenfurt entwickelte iterative Prozessmodell zu verwenden, das in fünf Phasen (Startphase, Iteration #2, Iteration #3, Iteration #4, Iteration #5) gegliedert ist.





4 Systemdokumentation

4.1 Lösungsweg

4.1.1 Gewählte Lösung HTL Entwicklungsboard mit PIC18f4550

Die größte Hürde dieses Projektes war das Einlesen der Daten. Dafür war der größte zeitliche Aufwand nötig, weil auch anfänglich einige Irrwege eingeschlagen wurden.

Die Daten wurden zuerst in das HTL-Entwicklungsboard mit dem PIC18f4550 über RS232 eingelesen. Dazu wurde zuerst das Signal vom Datenbus, der Bedienteil und Steuerung des LG500 (UZA) verbindet, von RS485- auf RS232-Pegel gewandelt.

Dann wurden jene Telegramme, welche die gewünschten Daten enthielten sozusagen herausgesiebt und von diesen Telgrammen wurde der Teil, der die Daten enthält in Variablen gespeichert. Da bei 16 Bit Daten eine Umwandlung notwendig war, um sie korrekt in zwei 8-bit Datenpakete als Zahl (integer) zu speichern, war da auch noch eine Programmlogik notwendig.

Nachdem es mit „Polling“ nicht wie gewünscht funktionierte, habe ich den Rat meines Zweitbetreuers befolgt und ein Programm geschrieben, welches mittels Interrupt immer dann ein Byte eingelesen hat, wenn es reingekommen ist und einstweilen alle anderen Routinen ruhen ließ. (Siehe 3.2.1 Sourcecode ab Seite 33).

Das Umwandeln der eingelesenen Parameter in einen ASCII-String und Rausschreiben über die zweite RS232-Schnittstelle mittels „fprintf“-Befehl dauerte nämlich zu lange und verursachte Probleme beim Polling-Programm. Da beim Interrupt-Programm das Einlesen der Bytes Priorität hatte, passierte es nicht mehr, dass der RS232-Einlesepuffer überlief und sich der PIC „aufhängte“.

Anschließend wurden die Daten dann mit „fprintf“ auf der zweiten RS232 Schnittstelle zum SBC65EC weitergeschickt. Dort wurden die Daten dann Variablen zugewiesen und auf der Webseite angezeigt.

Beim Kauf des SBC65EC wird bereits Software mitgeliefert, bei der der TCP/IP-Stack implementiert ist und mit der man die meisten Funktionen des SBC65EC bereits nutzen kann. Der Sourcecode der Software kann auch im Internet heruntergeladen werden, um ihn adaptieren zu können. Siehe Punkt 1.4.4 Modifikationen am SBC65EC ab Seite 15.

4.1.2 Alternative Lösungen

Zuerst wurde versucht die Original-Software, die auf dem Bedienteil läuft, jedoch für einen veralteten Microcontroller geschrieben wurde und von der der Compiler nicht bekannt war, auf dem HTL-PIC Board mit dem CCS-C- Compiler umzuschreiben und so zu adaptieren, dass das Protokoll mit dem HTL-PIC-Board nachgebildet werden würde.

Auf Grund der Komplexität des Protokolls, des unübersichtlichen und verwirrenden Programmierstiles des original Sourcecodes und weil erforderliche Datentypen und Methoden weder im CCS-C-Compilers noch im C18-Compiler, der für die Programmierung des HTL-PIC-Boardes zum Einsatz kam, vorhanden waren und nachdem selbst Professoren mit langjähriger PIC-Programmiererfahrung, die helfen wollten, dazu geraten haben, wurde dieser Weg aufgegeben.

Dann wurde versucht die Software, die mit dem Miniwebserver SBC65EC mitgeliefert wird so zu adaptieren, dass trotz der Serverfunktionen, die natürlich weiterlaufen mussten, auch noch die Daten der UZA einzulesen, zu filtern und dann auf der Webseite darzustellen.

Die Software auf dem SBC65EC stellte sich jedoch auch als sehr komplex heraus und ist mit dem C18-Compiler programmiert, was eine weitere Hürde darstellte, weil leider keiner der Professoren mit diesem Compiler Erfahrung hat und niemand weiter helfen konnte. Außerdem führte jede zusätzliche Funktion, die implementiert wurde dazu, dass der Miniwebserver stehen blieb, oder sich „aufhängte“.

Die umfangreiche mitgelieferte Software ist anscheinend auch so ziemlich die Grenze des Machbaren für diesen Microcontroller. Daher wurde dieses Vorhaben nach einigen Versuchen auch aufgegeben.

Der nächste Versuch wurde dann mit dem HTL-PIC-Board gestartet und mit dem bekannten und bereits verwendeten CCS-C-Compiler umgesetzt. Auf Anraten meines Projektbetreuers habe ich ein Programm geschrieben, welches die Daten mittels Bolling einliest.

Die Problematik beim Einlesen der Daten besteht darin, dass jedes ankommende Datenpaket eingelesen werden muss, da sich sonst der PIC aufhängt und nicht mehr weiterarbeiten kann. Gleichzeitig benötigt der PIC aber auch einige Rechenzeit für die Abarbeitung der Ausgabe, da auf der zweiten rausgeschriebene Strings erst in ASCII umgewandelt und zusammengesetzt werden müssen.

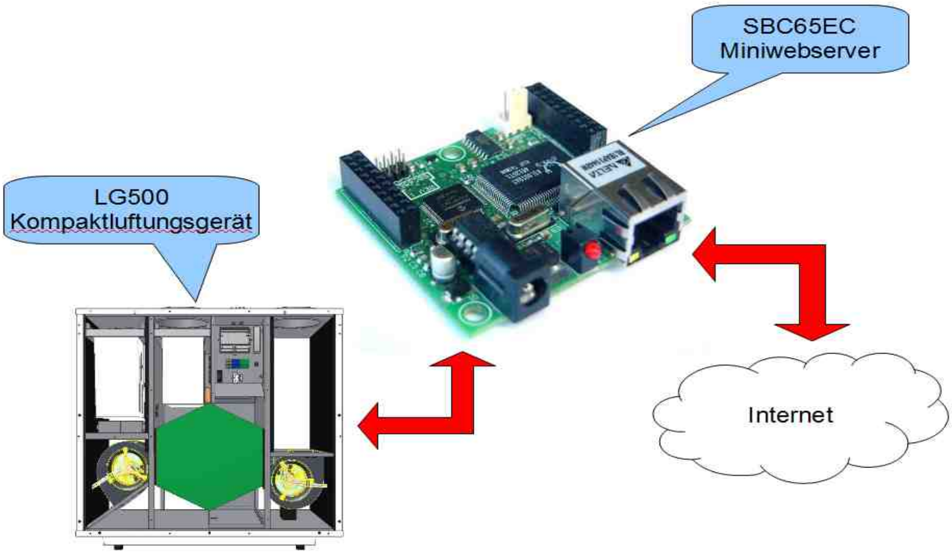
Daher entstand die Idee, eine kleine Routine zu schreiben, die prüft, ob Daten zum Einlesen vorhanden sind. Wenn ja, sollten sie eingelesen und gleich analysiert werden. Wenn nicht, sollte eine Ausgabe auf USART2 gemacht werden. Das jedoch auch nur dann, wenn sich Parameter verändert hatten.

Leider brauchte der PIC allem Anschein nach trotzdem immer wieder zu lange für die Ausgabe, sodass der Einlesepuffer der USART1, an der die UZA angeschlossen wurde mit Daten überflutet wurde und der PIC sich aufhängte.

Leider passierte das erst in einem Stadium des Programmes, als die Sache schon einige Wochen an Arbeitszeit verschlungen hatte und das Programm begann, recht komplexe Ausmaße anzunehmen. Schließlich wurde dieser Irrweg jedoch auch verworfen.

4.2 Grobentwurf

Folgendes Blockschaltbild entstand nach erstem Wissensstand:



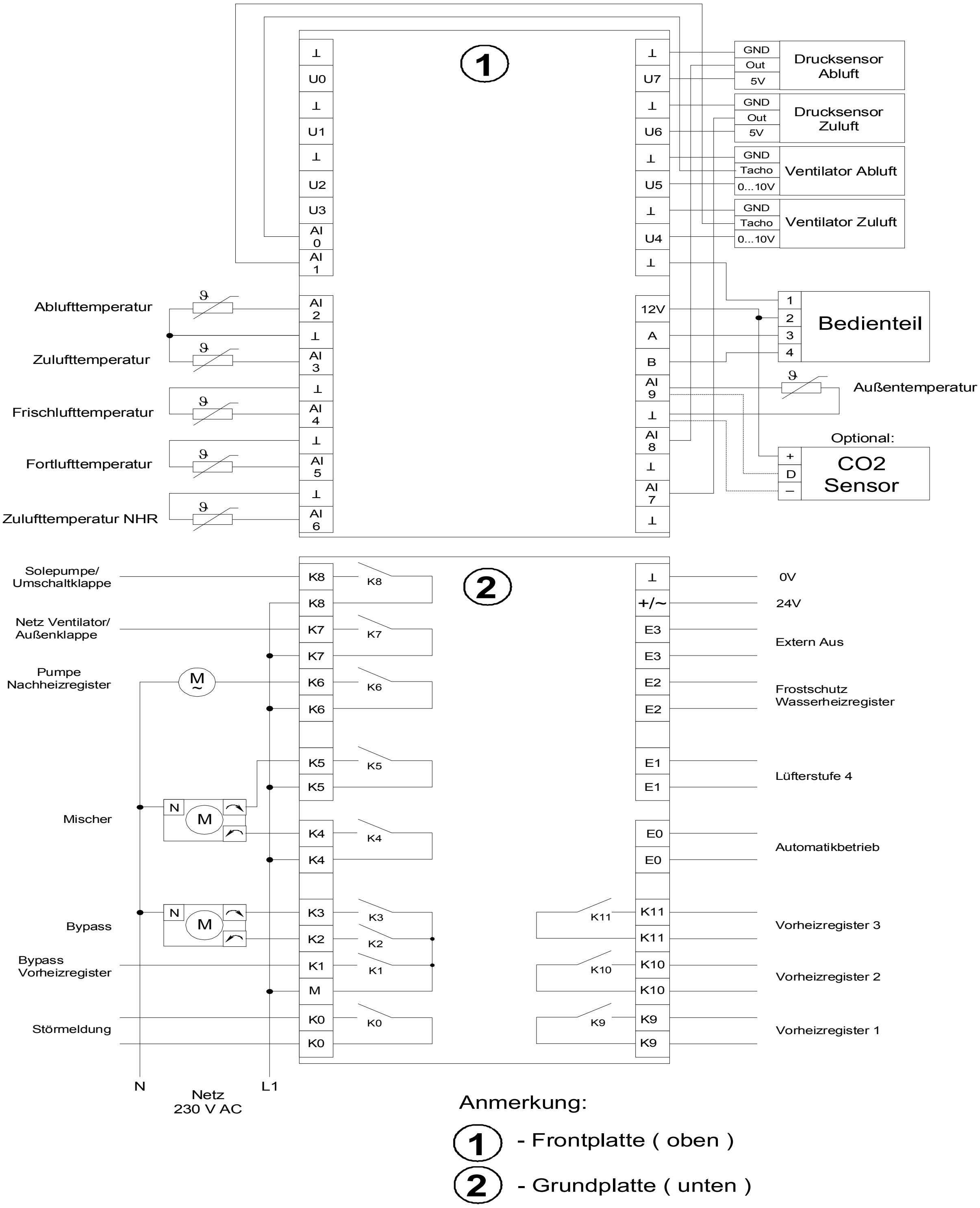
4.3 Feinentwurf

4.3.1 Verbindung aufbauen mit dem LG500 und Daten der Lüftungsanlage auslesen

Der Erste Schritt bestand also darin, sich mit der Steuerung des LG 500 auseinander zu setzen, um Möglichkeiten auszuloten, wie man eine Verbindung zu der Steuerung aufnehmen könnte. Dazu wurde die Betriebsanleitung des LG500 eingehend studiert. Auf der nächsten Seite finden Sie den Stromlaufplan der UZA (Steuerung des LG500) von Hermes Electronic.

Der Miniwebserver wird parallel zum Bedienteil angeschlossen. Damit wird der Miniwebserver mit den selben Daten versorgt, wie auch das Bedienteil, und die Kommunikation zwischen Bedienteil und LG500 kann abgehört werden. Diese Daten werden dann auf der Webseite des SBC65EC dargestellt.

Aus der Betriebs- und Montageanleitung des LG500 entnommener Anschlussplan der Steuerung UZA von Hermes Electronic:



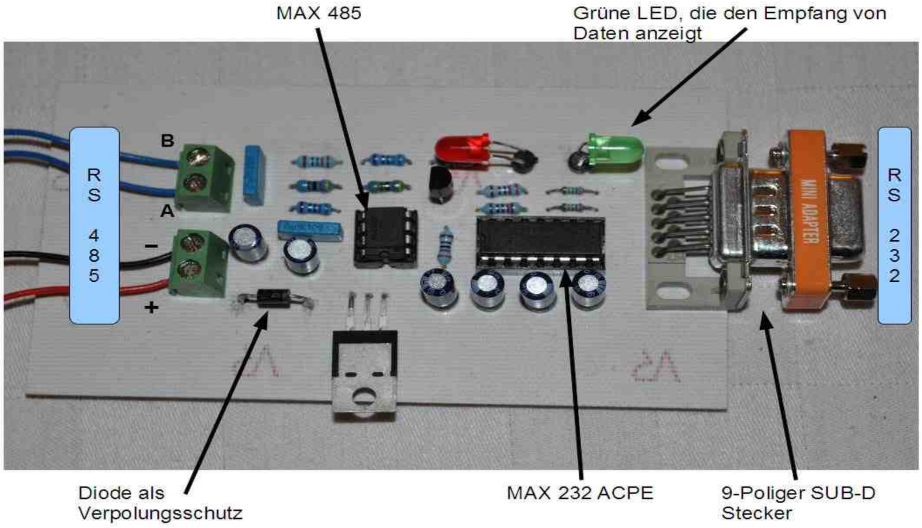
Nach eingehender Prüfung der Möglichkeiten habe ich mich entschieden, die Kommunikation zwischen Bedienteil und der UZA Steuerung zu nutzen, und meine Entwicklung parallel zum Bedienteil anzuschließen, da die Kommunikation zwischen UZA und angeschlossenem Bedienteil sämtliche Daten liefert, die auf der Webseite dargestellt werden sollen.

4.3.2 Ausgelesene Daten sortieren, verarbeiten und an den Miniwebserver weiterleiten

Da die Verbindung zwischen UZA und Bedienteil ein RS485 Bus ist und das zu verwendende PIC-Board nicht über RS485, sondern nur über RS232 verfügt, und außerdem die Programmierung von RS232 auf dem HTL-PIC-Board viel einfacher umzusetzen war, habe ich mich entschieden, einen Pegelumsetzer von RS485 auf RS232 einzubauen, um dann alle Daten, die auf diesen Leitungen laufen, in das HTL-PIC-Board einzulesen.

Nach eingehender Recherche und da solche Datenconverter im Internet zu Preisen zwischen 20 und 1000 Euro zu finden waren, wurde zuerst eine selbst gefertigte und bestückte Platine erfolgreich eingesetzt.

Siehe Bild unten:

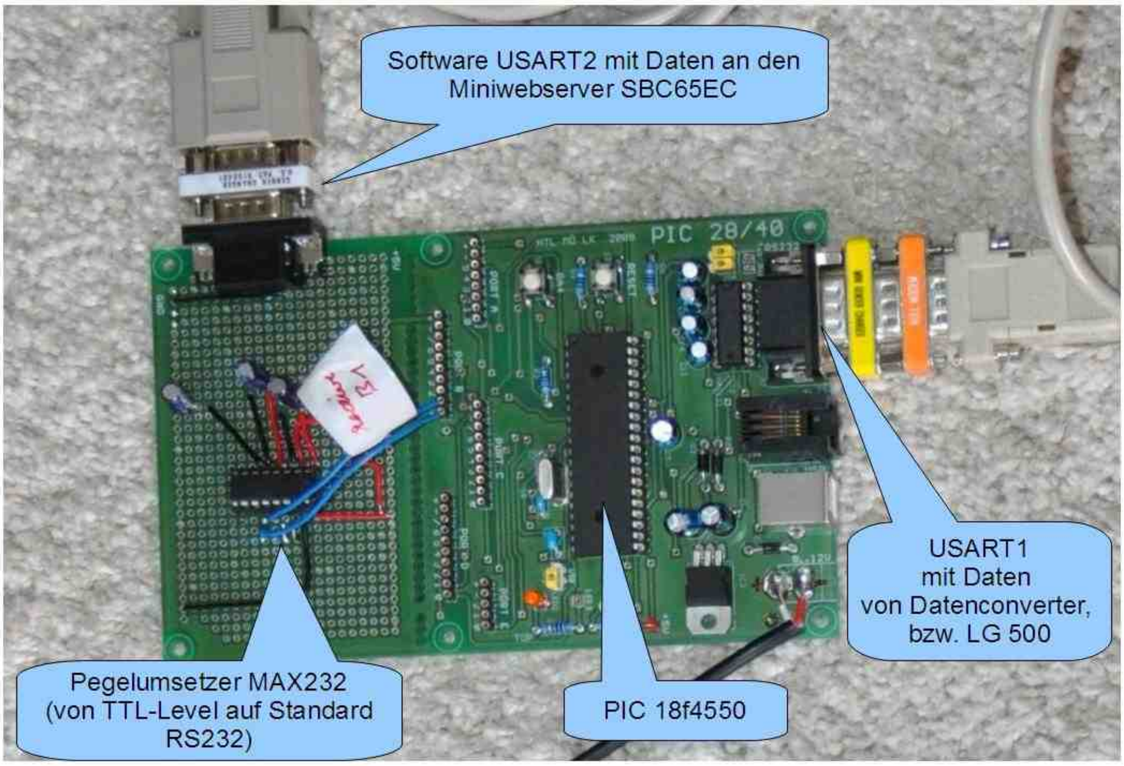


Aus Kosten- und Zeitgründen wird jedoch in der Serienfertigung ein zugekaufter Datenconverter von RS485 auf RS232 zum Einsatz kommen, der die Anforderungen genauso gut erfüllt. Siehe untenstehendes Bild:



Ausführliche Tests müssen jedoch erst noch zeigen, ob dieser Converter für die Serienproduktion des Miniwebservers und für diese Anwendung dauerhaft geeignet ist.

Zum Einlesen, Sortieren, Extrahieren und Weiterleiten der Daten kommt das HTL-PIC-Board zum Einsatz. Siehe Bild untenstehend:



Da das HTL-PIC-Board standardmäßig nur mit einer USART-Schnittstelle für RS232-Datenverkehr konzipiert wurde, habe ich eine Software-USART mit den Portpins B0 (Transmit (TX)) und B1 (Receive(RX)) implementiert. Dann habe ich für eine funktionsfähige RS232 einen MAX232-Pegelumsetzer aufgebaut.

Auszug aus einer Erklärung der Funktionsweise des Pegelumsetzers MAX232:

[...



„Der MAX232 ist einer der bekanntesten Pegelkonverter für die RS232 (serielle Schnittstelle des PCs). Es ist häufig der Fall, dass nur TTL Pegel (0 und 5V) zur Verfügung stehen, aber trotzdem mit dem PC nach dem RS232 kommuniziert werden soll. Die wenigsten Geräte verfügen über die geforderte Spannung von +12V und -12V. Eine eigene Spannungsversorgung nur für die RS232 wäre ebenfalls ein zu hoher Aufwand. Abhilfe schafft hier der MAX232. Diesem genügt eine 5V-Versorgungsspannung, um eine Wandlung von TTL auf RS232-Pegel vorzunehmen. Die benötigten +12V und -12V werden von dem IC

selbst erzeugt. Als externe Beschaltung genügen 4 Elkos mit ca. 1µF. Außerdem wird der Rest des Geräts gegen ESD Beschädigung über die Schnittstelle geschützt.

Verwendung findet der MAX232 zum Beispiel in Notebooks, Modems, in Mikrocontrollerschaltungen, die eine RS232 benötigen,..

Zur Funktion

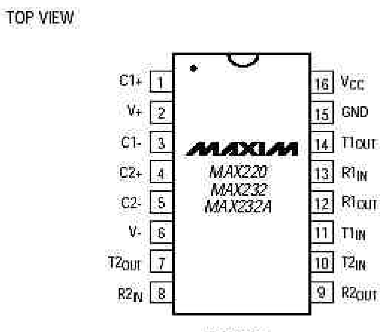
Der MAX232 verfügt über 2 integrierte DC-DC-Wandler, einer zur Spannungsverdopplung (+10V) und einer als Spannungsinverter (-10V). Für diese Wandler werden die 4 Kondensatoren benötigt.
Um eine korrekte Wandlung eines TTL Signals auf ein RS232 Signal zu ermöglichen, muss eine Pegelumsetzung auf die höheren Pegel erfolgen und zusätzlich muss das Signal invertiert werden.
Da eine logische 1 bei TTL 5V und bei RS232 -12V entspricht.
Die Wandlung von RS232 auf TTL erfolgt analog zu oben, ein Pegel von -12V wird auf 5V umgesetzt.

Der MAX232 verfügt über zwei TTL->RS232 und RS232->TTL Stufen. Eine davon wird meist für RxD und TxD verwendet. Die zweite kann entweder für Steuerleitungen der RS232 oder für einen zweiten Kanal verwendet werden.“..]

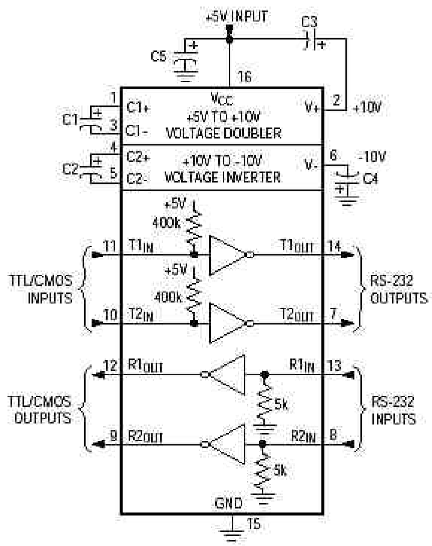
[LEW 07]

Untenstehender Beschaltungsplan aus der Dokumentation des MAX232

kam zum Einsatz:



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

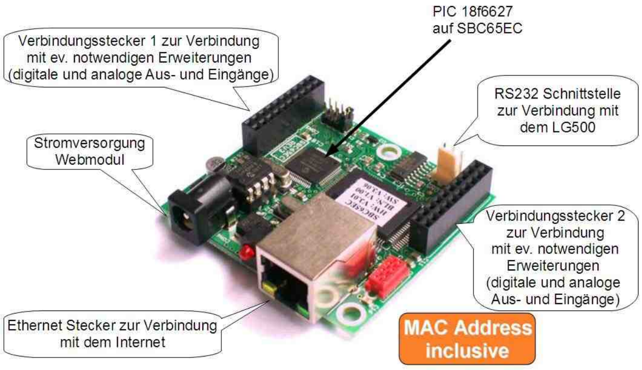


Nach eingehender Analyse des Datenverkehrs zwischen UZA-Steuerung und dem Bedienteil stellte sich heraus, dass ca.120 Parameter zyklisch mit einem Abstand von ca. 10 ms hin- und hergeschickt werden. Die UZA verwendet dabei eine Datenrate von 19200 Baud. Ein Telegramm beginnt immer mit der Zahl 04 und endet immer mit der Zahl 05. Daher ist es

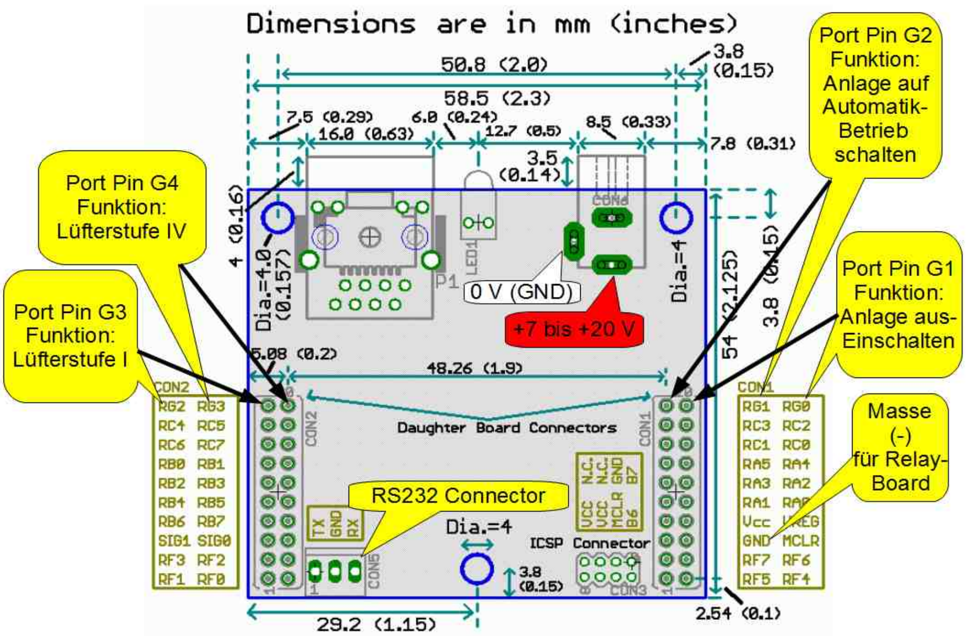
relativ einfach Anfang und Ende der Telegramme beim Einlesen für die weitere Verarbeitung der Telegramme zu finden und zu verwenden. Die einzelnen Parameter sind gekennzeichnet durch eine ID an einer bestimmten Stelle der Telegramme. Daher war es möglich nur die Parameter, die für die Anzeige auf der Webseite gebraucht wurden, herauszufiltern. Mehr dazu später.

4.3.3 Daten der Lüftungsanlage auf der Webseite des Miniwebservers darstellen

Zum Einsatz kam bei diesem Projekt ein SBC65EC Miniwebserver der Firma Modtronix. Siehe Bild untenstehend:



Untenstehend der Beschaltungsplan des SBC65EC Miniwebservers:



4.3.4 Modifikationen am zugekauften Miniwebserver SBC65EC:

Der Miniwebserver wird mit funktionierendem TCP/IP Stack und diversen Funktionen geliefert. Um jedoch Daten über die RS232-Schnittstelle einlesen zu können und diese korrekt auf der Webseite darstellen zu können, waren einige Modifikationen notwendig.

Die Datei cmd.c musste ergänzt werden.

Ich führte eine neue „Value Tag Gruppe“, mit Namen „H“ ein, nutzte bereits implementierte Routinen, um die Daten zu bekommen und schrieb byteweise in die Stringvariable strTmp. Bis zum Ende des Strings, der mit „\0“ gekennzeichnet war.

Sourcecode:

```
// Handle MyOwn value tags - key is 'H'
  else if (tagGroup == 'H')
      {
      // Parse tag and value and fill-in text
      if (ref == HTTP_START_OF_VAR)
      {
       cmdGetMyOwnValue(tagVal, strTmp);
       }
      *pGetTagInfo->val = strTmp[(BYTE)ref];

      if (strTmp[(BYTE)ref] == '\0')
      {
       pGetTagInfo->ref = HTTP_END_OF_VAR;
       }
      else
         {
          pGetTagInfo->ref++;
           }
      return 1;   //One byte was written, exit
      }

  //////////////////////////////////////////////////
```

Dann wurden in cmd.c die eingelesenen Werte in die entsprechenden „Tags“ geschrieben:

```
void cmdGetMyOwnValue(BYTE tagVal, BYTE* strRes)
     {
      strRes[0] = '\0';        // just in case
      switch (tagVal)
         {
           case 0:     sprintf(strRes,"%s", usartString); // Received string
               break;
           case 1:     sprintf(&strRes[0],"%s", &rlf[0]); // Filter Restlaufzeit
               break;
```

```
            case 2:     sprintf(&strRes[0],"%s", &lst[0]); // Lüfterstufe
                break;
            case 3:     sprintf(&strRes[0],"%s", &srt[0]); // Sollwert
                break;
            case 4:     sprintf(&strRes[0],"%s", &irt[0]); // Istwert
                break;
          }
      }
```

In der Datei „mxwebsrvr.c“, dem Hauptprogramm wurden neue Variable eingeführt:

```
////////////////////////////////////////////////////
//Globale Variable für die Parameter
 char rlf[10]; //Restlaufzeit Filter
 char lst[10]; //Lüfterstufe
 char srt[10]; //Sollwert Raumtemperatur
 char irt[10]; //Ist-Wert Raumtemperatur
  int i = 0;    //Counter für for Schleife (loop)
  int StringIndex = 0; //Index for Strings;


/////////////////////////////////////////////////
```

Die eingelesenen Strings wurden in die entsprechenden Variablen geschrieben:

```
////////////////////////////////////////////////////////////////////////////////////
///
         //Handling für Einlesen von USART1

      if (serIsGetReady()) // Something in receive buffer ?
      {
     serbyte = serGetByte(); // Fetch byte received
     usartString[index] = serbyte;
     if (index < (sizeof(usartString)-1)) index++;
    usartString[index] = 0;
         if (serbyte == '.' || serbyte == '\r')
              {
            usartString[(index-1)] = '\0';   //terminates the String
               stringComplete = 1;

               index = 0;            //Reset the Index for the String we are
reading into
          }
       if (stringComplete == 1)
          {
           i=0;

              switch(usartString[0]) // The id information is in this position of
incoming string
              {
                 case '1': //Filter Restlaufzeit

                    while (usartString[i] != '\0')
                    {
                    rlf[i] = usartString[(i+1)]; //the other controller sends this:
```

```
                    i++;                            //fprintf(WEB, "1%Lu.",rlf);
//Rest- Laufzeit Filter
                    }                          //so we are getting 4 ASCII numbers
                   i = 0;                      //Reset i after writing into rlf
                break;

                case '8': //Lüfterstufe

                    while (usartString[i] != '\0')
                       {
                       lst[i] = usartString[(i+1)]; //fprintf(WEB, "88%u\r",lst);
                     i++;
                      }                          //so we are getting 2 ASCII numbers
                      i = 0;
               break;

               case '7': //Sollwert Raumtemperatur

                    while (usartString[i] != '\0')
                       {
                       srt[i] = usartString[(i+1)]; //fprintf(WEB, "7%u.",srt);
                     i++;
                       }                        //Sollwert Raum- Temperatur(p6))
                                         //so we are getting 2 ASCII numbers
                    i = 0;
               break;

               case '6': //Istwert Raumtemperatur

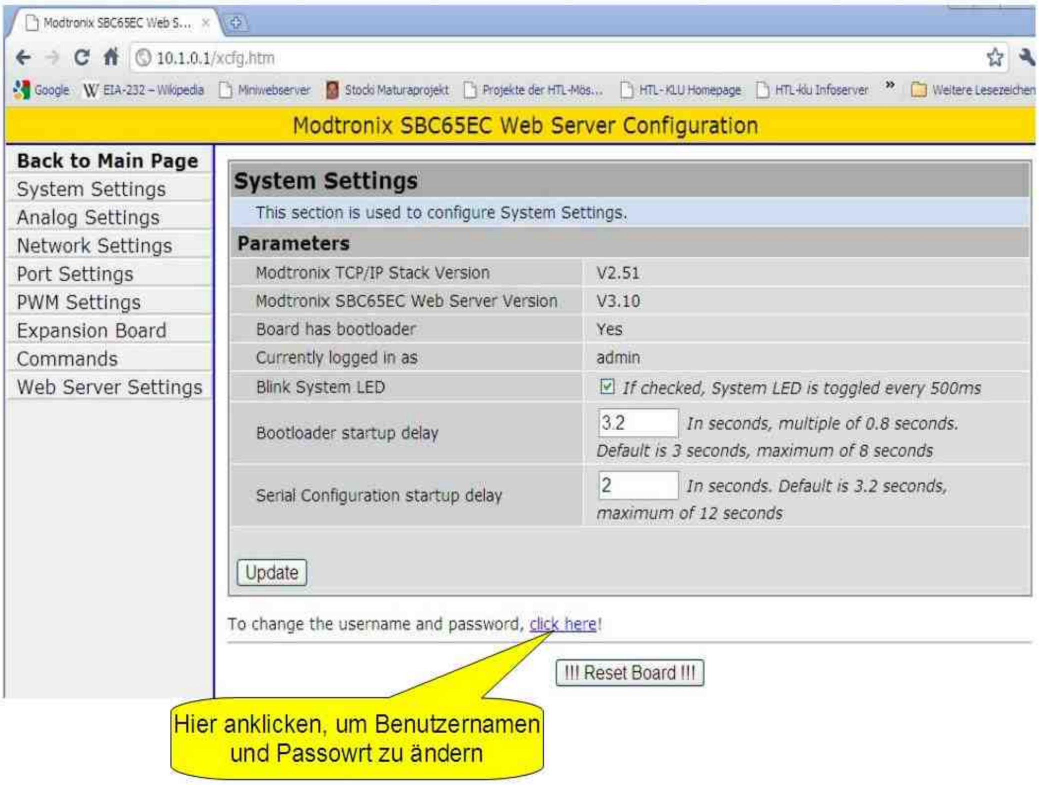
                    while (usartString[i] != '\0')
                       {
                       irt[i] = usartString[(i+1)];// fprintf(WEB,
"6%Ld.",irt);   // Ist-wert Raum- Temperatur
                       i++;
                      }                          //so we are getting 4 ASCII numbers
                   i = 0;
               break;
              stringComplete = 0;   //Reset the string complete flag
              }    //Ende switch (usartString[0])
           }      //Ende if (string complete == 1)
      }             //Ende if (usartbool == 1)
```

Schließlich wurde noch die Webseite selbst ein wenig umgebaut. Sämtliche Funktionen wurden behalten, aber nur im Hintergrund, weil der Kunde diese ohnehin nicht benötigt.

Aus der index.htm wurden einfach die Links zu den Konfigurationsseiten herausgelöscht. Damit sind diese Seiten trotzdem noch erreichbar, wenn man sie in das Adressfeld des Browsers eingibt, aber für den Kunden durch einfaches Anklicken nicht mehr präsent.

Die auf diese Weise im Hintergrund versteckten Webseiten sind nur für den Servicetechniker interessant, um den Miniwebserver zu konfigurieren.

Um das Passwort und den Benutzernamen zu ändern gibt man im Adressfeld des Browsers zusätzlich zur IP-Adresse des Miniwebservers */xcfg.htm* ein, klickt auf den Button „Web Server Settings“ und gelangt auf folgende Seite:

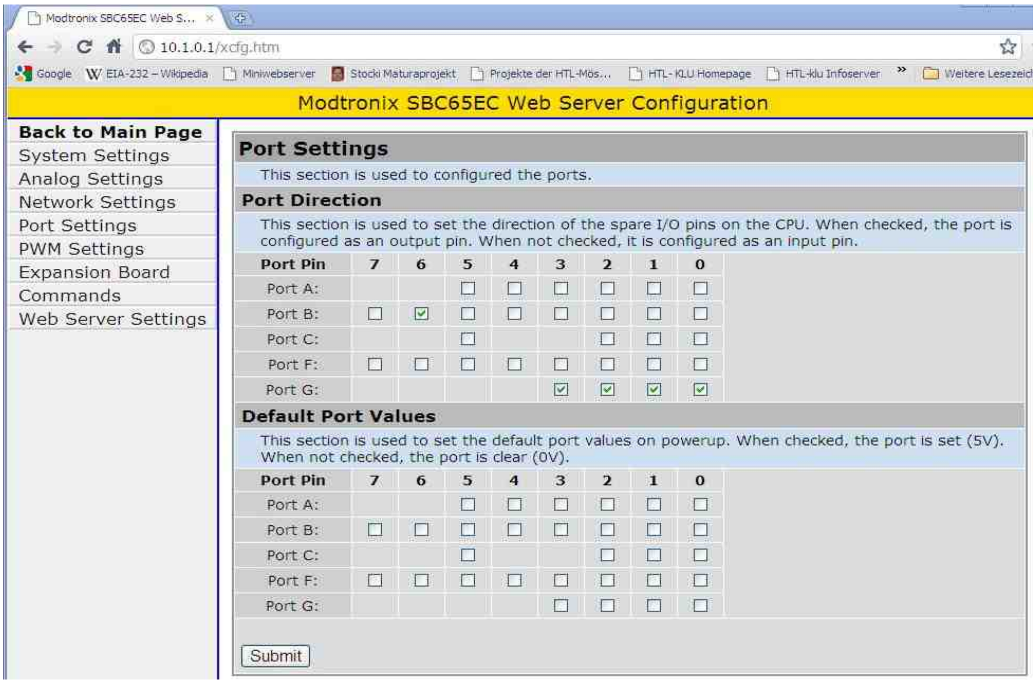


Damit gelangt man dann zur Eingabe des Benutzernamens und des Passwortes:



Man gibt den neuen Benutzernamen und das Passwort ein und klickt auf „Submit“.

Durch Anklicken des Buttons „Port Settings“ gelangt man auf folgende Seite:



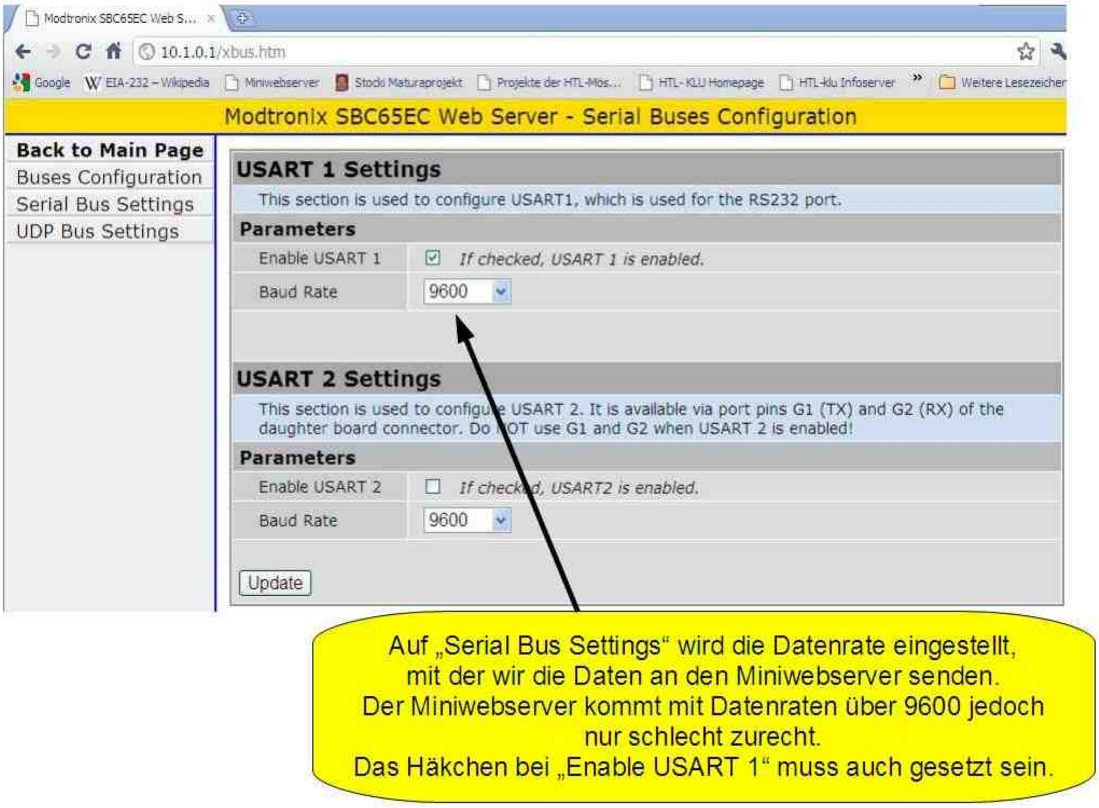
In unserem Fall haben wir den Port G für unsere Applikation genutzt und die Häkchen bei Port G müssen gesetzt werden.

Wenn man mehr Funktionen über die Webseite schalten möchte, könnte man noch Port F (7), Port C (4), Port B (7) und Port A (5) durch Anklicken der Häkchen auf dieser Seite zu digitalen Ausgängen machen. D.h. mit einer entsprechenden Erweiterungsplatine, wo dann mehr Relais aufgebaut und mit den Portpins verbunden sind, könnten dann theoretisch bis zu 27 Relais über die Webseite durch Anklicken der entsprechenden Kästchen geschaltet werden.

Für zukünftige Erweiterungen gibt es also noch genügend Reserven, zumal man auf der UZA (Steuerung des LG500) noch weitere digitale Eingänge mit Hilfe der Parametrisierungssoftware freischalten und Funktionen für diese digitalen Eingänge festlegen könnte. Für jeweils 4 weitere Funktionen würde dann ein zusätzliches CAT5 Kabel benötigt werden.

Die zweite versteckte Seite, die der Servicetechniker braucht, ist die Seite, auf der die seriellen Einstellungen auf dem Miniwebserver eingestellt werden.

Man schreibt nach der IP-Adresse des Miniwebservers im Eingabefeld des Browsers */xbus.htm* und gelangt auf folgende Seite:



Dann wurde die Seite umgebaut, auf der man normalerweise sämtliche Ports des Webservers durch Anklicken schalten kann.

Die Datei ioval.cgi wurde wie untenstehend verändert:

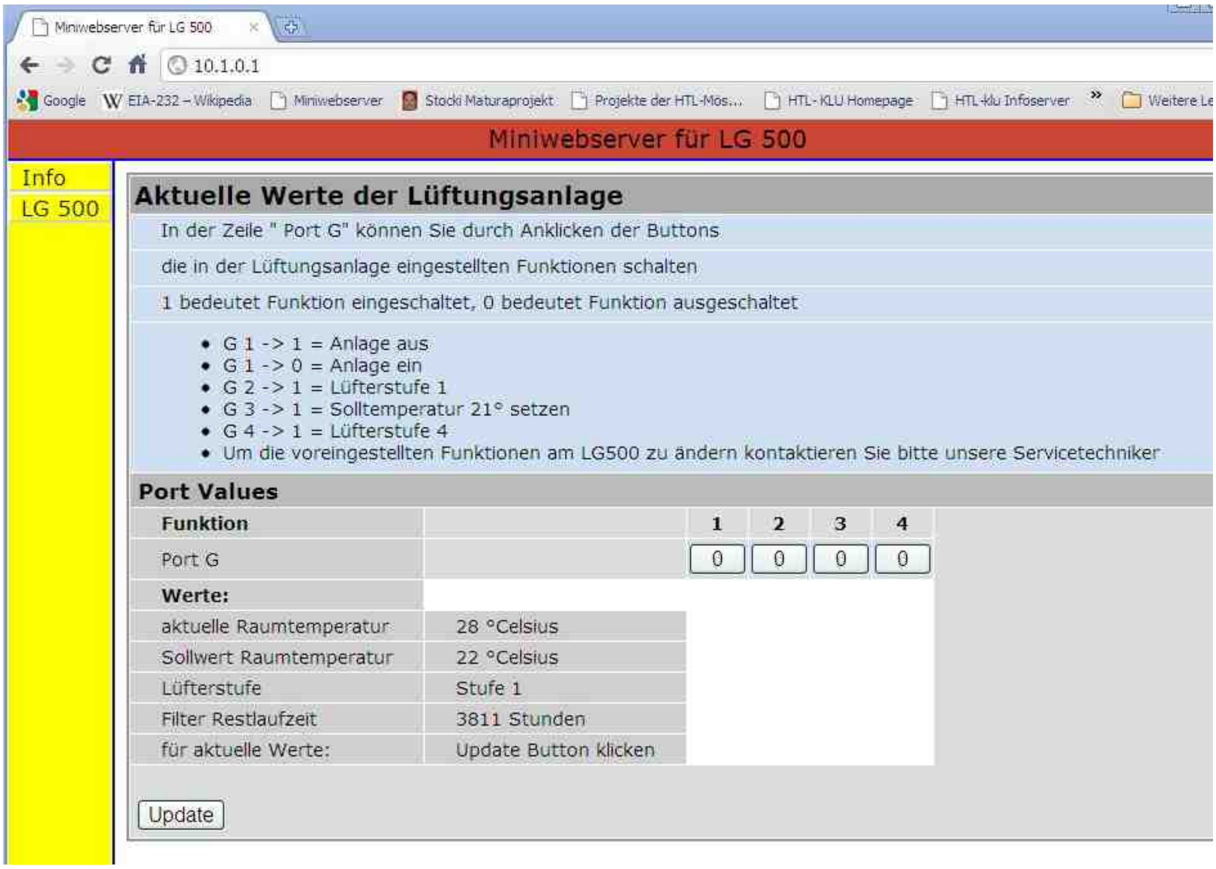
```
<html>
<head>
<meta http-equiv="Pragma" content="no-cache">
<link href="mx.css" rel="stylesheet" type="text/css">
<script src="lib01.js"></script>
<script type="text/javascript">
function lnb() {
        document.write("<td class=bCel> </td>");
}
function ln(io,val,name) {
        document.write("<td class=bCel align=center style=\"padding:1px;\">");
        if(io==0) {
                document.write("<input style=\"width:38px;\" type=submit
value="+val+" name=\""+name+"\">");
        }
        else {
                document.write(val);
        }
        document.write("</td>");
```

```
}
function tdh(s) {
        document.write("<td class=bCel align=center><b><div
style=\"width:30px;\">"+s+"</div></b></td>");
}
</script>
</head>
<bodyclass=ifrmBody>
<form method=GET action=IOVAL.CGI>
<table class=bBox cellpadding=3 cellspacing=1>
<tr><td class=bHdr colspan=10>Aktuelle Werte der Lüftungsanlage</td></tr>
<tr><td class=bDesc colspan=10>Hier können Sie die voreingestellten Funktionen
schalten
<ul><li>Sie können immer nur eine Funktion auf einmal schalten</li>
<li> 1 bedeutet eingeschaltet, 0 bedeutet ausgeschaltet</li>
<li>Um die voreingestellten Funktionen am LG500 zu ändern</li>
<li>kontaktieren Sie bitte unsere Servicetechniker</li> </ul>
</td></tr>
<tr><td class=bSec colspan=10>Port Values</td></tr>
<tr>
        <td class=bLbl><b>Funktion Nummer</b></td>
        <script type="text/javascript">
        tdh(" ");
        tdh("1");
        tdh("2");
        tdh("3");
        tdh("4");
        </script>
        <td class=bCtr rowspan=6 width=100%%></td>
</tr>
<tr>
        <td class=bLbl>Port G</td>
        <script type="text/javascript">
        lnb();
        ln(%g50,%g00,"gx0");
        ln(%g51,%g01,"gx1");
        ln(%g52,%g02,"gx2");
        ln(%g53,%g03,"gx3");
        lnb();
        lnb();
        lnb();
        </script>
</tr>
<tr>
     <br>aktuelle Raumtemperatur</br>
         <br>
                 %H01
        </br>
</tr>
<tr>
     <br>Sollwert Raumtemperatur</br>
         <br>
                 %H02
        </br>
</tr>
<tr>
         <br>Lüfterstufe</br>
         <br>
                 %H03
         </br>
</tr>
<tr>
         <br>Filter Restlaufzeit</br>
         <br>
                 %H04
         </br>
         <br>  </br>
</tr>
```

```
<tr>
         <br>Data received</br>
         <br>
                 %H00
         </br>
         <br>  </br>
</tr>
<tr><td class=bBot colspan=10><input type=submit value="Update"></td></tr>
</table>
</form>
</body>
</html>
```

Dabei sind %H00, %H01 usw. Platzhalter („Tags“) für die eingelesenen Werte in mxwebsrvr.c

Die Seite, auf der die aktuellen Werte angezeigt werden und auf der man auch die eingestellten Funktionen der Lüftungsanlage schalten kann sieht wie folgt aus:



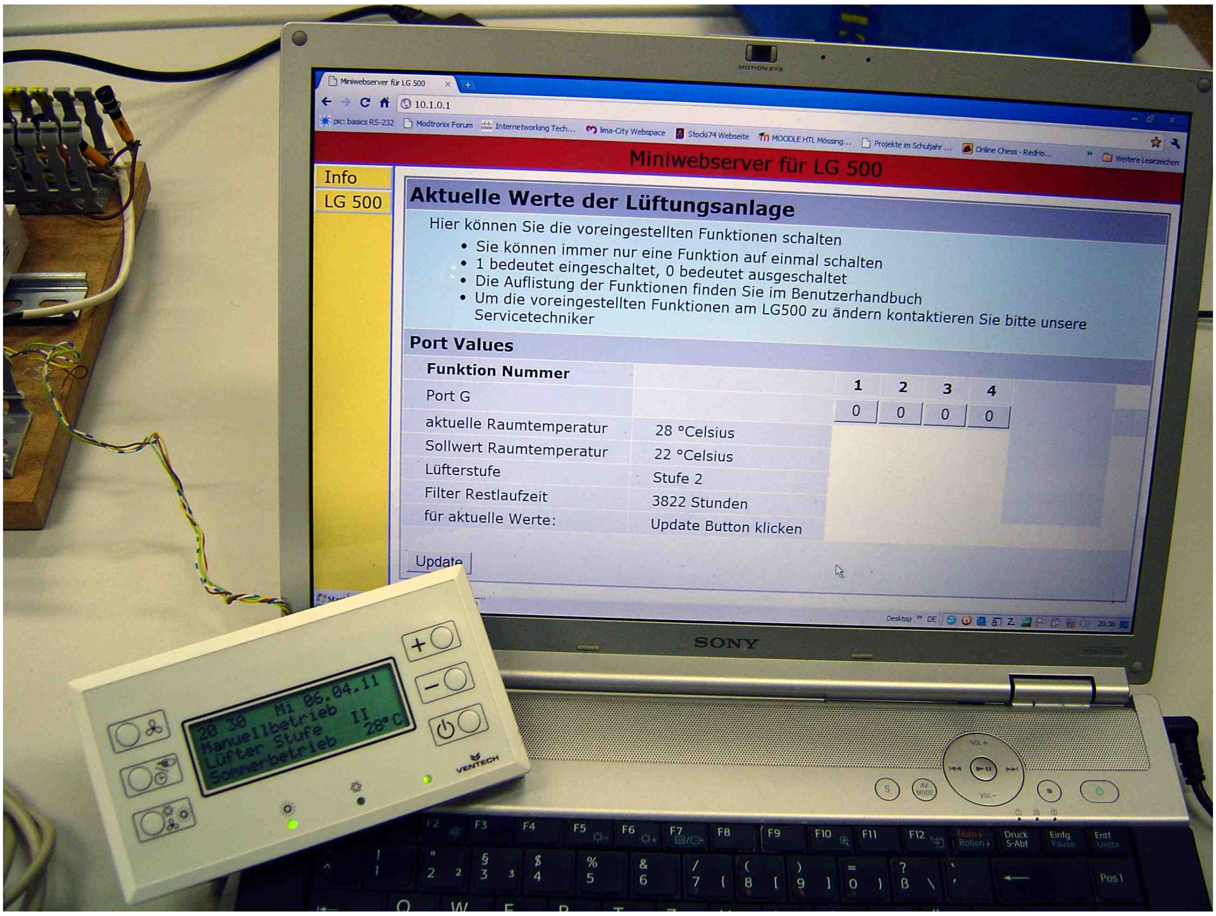
Um aktuelle Werte angezeigt zu bekommen, müssen diese Daten erst einmal über den Datenbus, der das Bedienteil mit dem LG500 verbindet, gesendet werden.

Der Miniwebserver registriert jede Änderung der Werte. D.h. dass der Miniwebserver erst dann aktuelle Daten anzeigen kann, wenn sich auf dem Display des Bedienteiles diese Werte ändern.

Durch Klicken des „update“ Button werden die aktuell im Miniwebserver angekommenen und gespeicherten Werte abgerufen und auf der Webseite angezeigt.

Untenstehend ist ein Foto der funktionierenden, fertigen Webseite:

Zu sehen ist, dass die Daten, die auf der Webseite angezeigt werden gleich sind, wie die, welche sich auf dem Display des Bedienteiles finden.



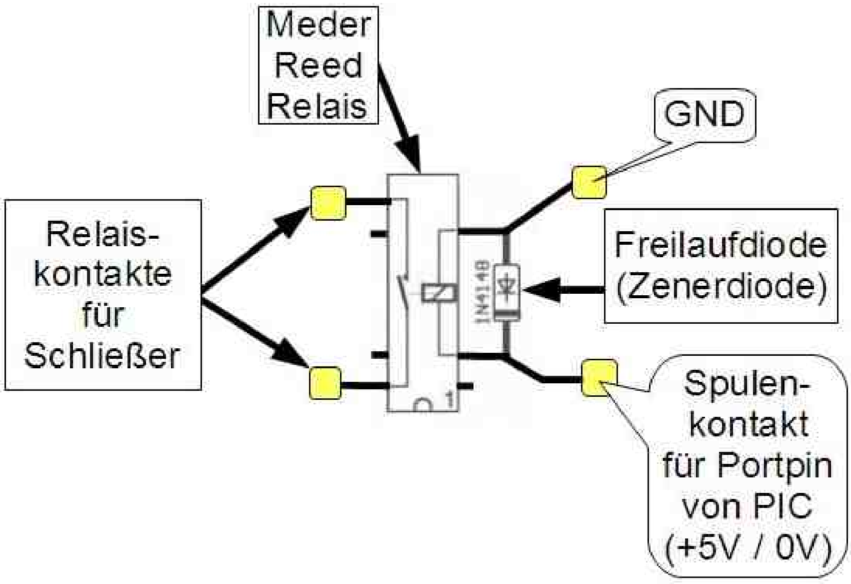
Des Weiteren ist zu beachten, dass die Werte aktualisiert werden, wenn der Benutzer auf den Button „Update“ klickt. Die vordefinierten Funktionen des LG500 lassen sich durch Anklicken der Buttons in der Zeile „Port G“ schalten. Des Weiteren sind einige für Kunden interessante Informationen zu finden, wird der Button „Info“ angeklickt.

4.3.5 Steuerbefehle des Benutzers von der Webseite (Internet) an das LG500 übermitteln und diverse Funktionen am LG500 schalten

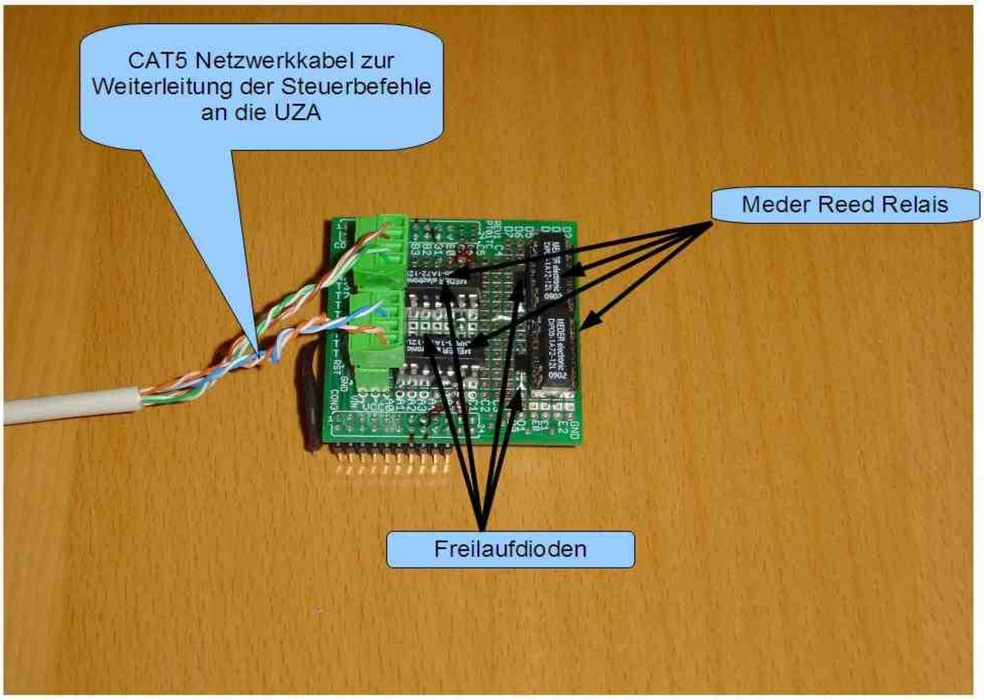
Um Steuerbefehle an die SBC65EC zu senden und weil dafür das Durchschalten von Spannungen mit ca. 12 Volt notwendig war, wurde der SBC65EC mit einer Entwicklungs-Lochrasterplatine erweitert, auf die 4 Meder-Reed-Relays aufgebaut wurden. Somit kann man durch Einschalten der Portpins des Miniwebservers über die Webseite (Port G)

konfigurierbare Funktionen bei der UZA auslösen. Die beiden Verbindungsstecker des SBC65EC kommen dabei zum Einsatz.

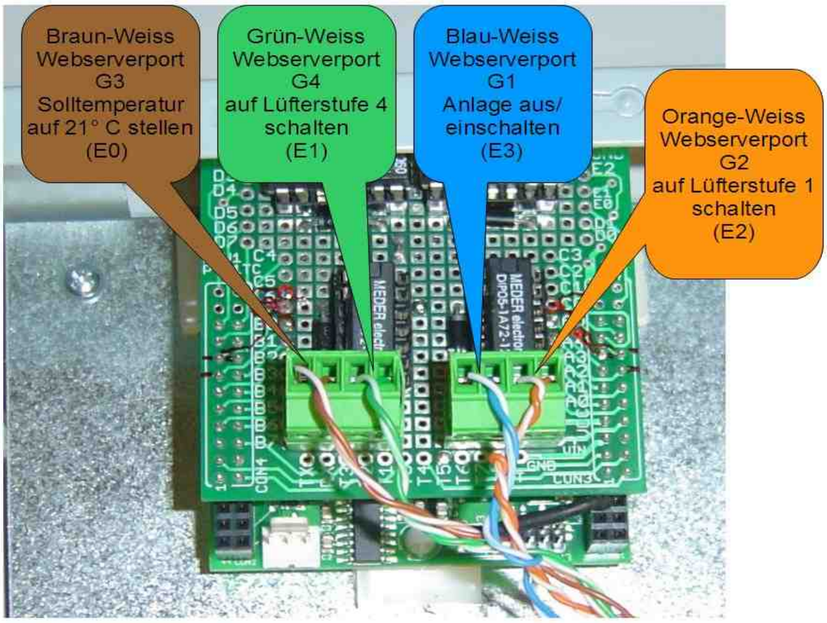
Beschaltungsplan des Meder-Reed-Relais:



Untenstehend das Bild der Erweiterungsplatine mit den Relays:

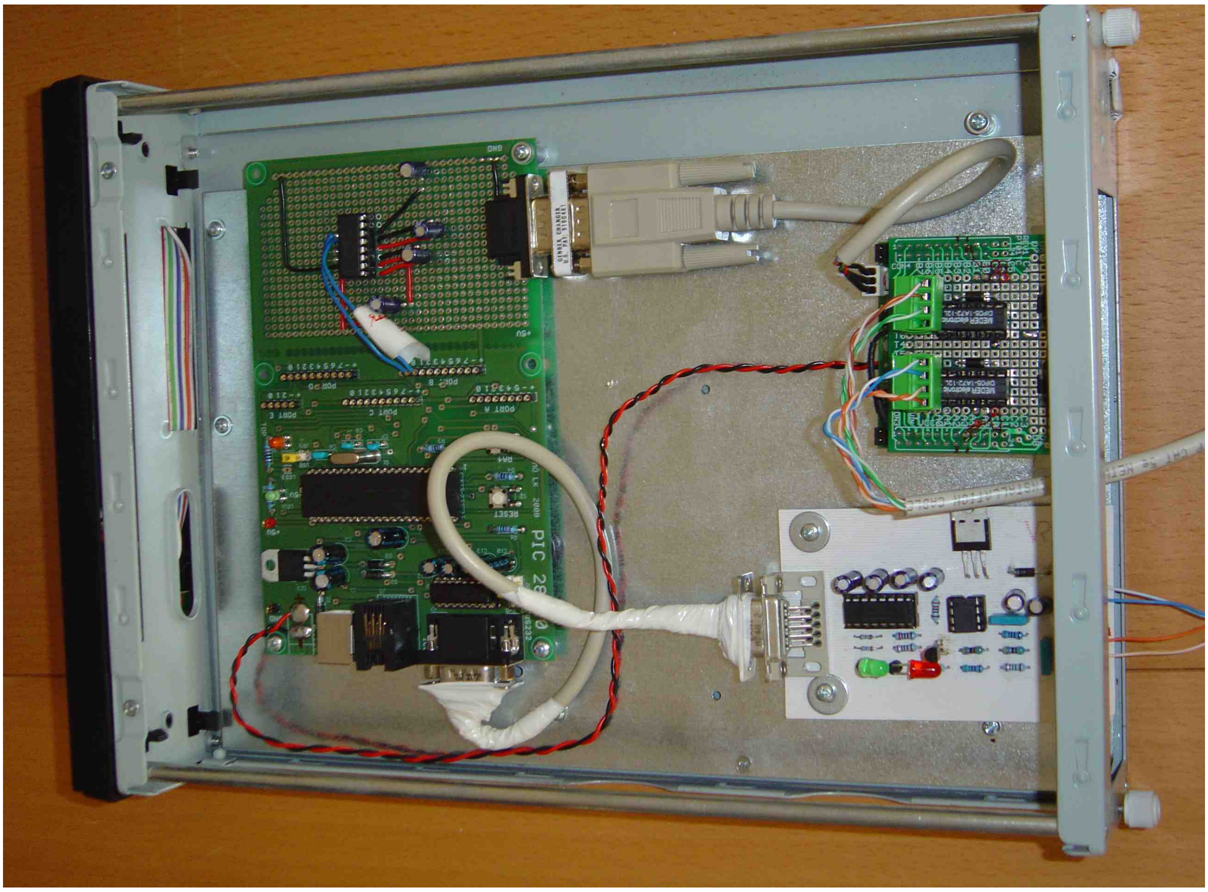


Beschaltungs- und Anschlussplan der Relais-Erweiterungsplatine:

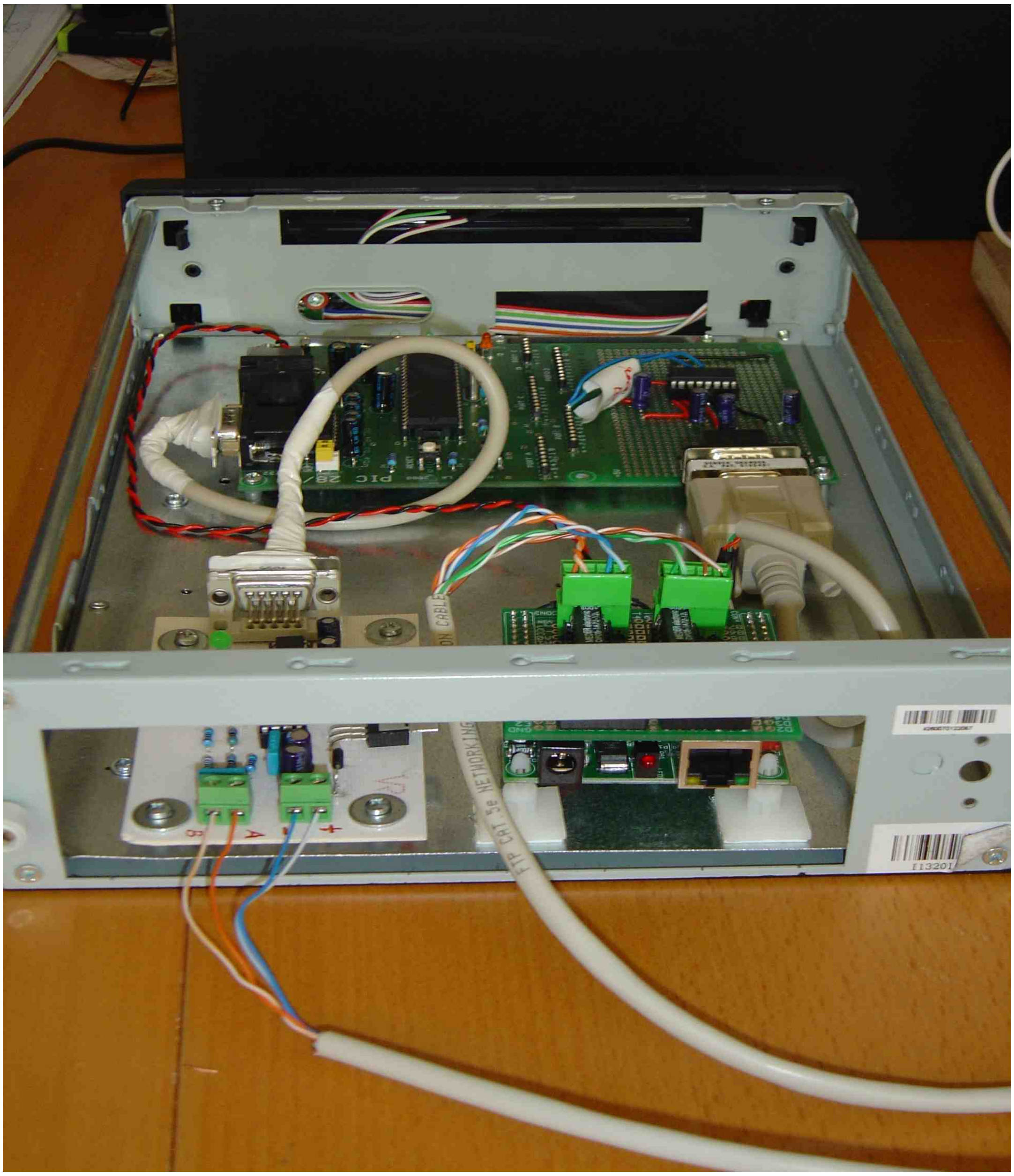


SBC65EC mit aufgesteckter Relais-Erweiterungsplatine fertig eingebaut in Mini-ITX Servergehäuse zusammen mit Datenconverter und HTL-PIC-Board:

Draufsicht:



Ansicht von Hinten:



4.4 Implementierung

4.4.1 Sourcecode

Untenstehend das finale funktionierende Interrupt-programm für das HTL-PIC Entwicklungsboard mit PIC 18f4550:

```
//                    Projekt Miniwebserver für LG500
//                    Copyright: Robert Stocker 2011
//                    7/8 ABELI 2010/2011
//
```

```
// ----------------
//
//     Compiler:  CCS PCWHD
//     Hardware:  HTL-Board neu mit HTL-BootLoader
//                Prozessor:  PIC 18F4550  48MHz (20MHZ-Quarz mit PLL)
//
//     Über 1. RS232 Stream (UZA) einlesen und über 2. RS232 Stream
//     (WEB) ausgeben
//
// Funktion:Dieses Programm liest Daten von einem Stream (UZA) in die Hardware
//          USART ein, analysiert diese Daten (Bytes) und schreibt sie
//          in einem anderen Stream (WEB) byteweise über Software USART raus
//          Auf dem HTL-Board wurde dazu ein MAX232 zur Pegelumsetzung von
//          TTL-Level auf RS232 aufgebaut und mit den Pins B0 (Receive) und
//          B1 (Send-Transmit) verbunden (Siehe CCS-Help-Doku)
//
//
//  PIC 18F4550 (RS232) und CCS-Compiler
//
// uC-Einstellungen
----------------------------------------------------------------------------
#include <18F4550.h>
#device ADC=10
//#fuses  HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN      //
nicht verändern
//#use    delay(clock=48000000)                                  // nicht verändern

#fuses  HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,               // nicht verändern
#use    delay(crystal=20000000)                                  // nicht verändern
#fuses ICSP1

// RS232
-----------------------------------------------------------------------------------
-------
#use    RS232(baud=19200, xmit=PIN_C6, rcv=PIN_C7, stream=UZA)
#use    RS232(baud=19200, xmit=PIN_B0, rcv=PIN_B1, stream=WEB)

#include <string.h>

#define SOT 4   //Start of telegram
#define EOT 5   //End of telegram

// globale Variable
unsigned int     In_Char = 0;     //eingelesenes Zeichen
volatile char RS232_Telegram[40]; //String, in den Telegramme eingelesen werden
volatile unsigned int   RS232_Index = 0;//Index für String
volatile int     T_id = 0;             //Telegramm ID
volatile short   Rs232_Receive = 0;    //Sind wir am Anfang eines Telegrammes? SOT?
volatile int     Auswertung_ok = 1;    //Haben wir ein gewünschtes Telegramm?

volatile int       rlf_H=0;      //High Byte von Restlaufzeit Filter
volatile int       rlf_L=0;      //Low Byte von Restlaufzeit Filter
volatile long       rlf=0;        //Restlaufzeit Filter (16 Bit Integer)
volatile int       srt=0;         //Sollwert Raumtemperatur (8 Bit)
volatile signed long irt = 0; //Ist Wert Raumtemperatur (16-Bit mit Vorzeichen)
volatile int      lst = 0;    //Lüfterstufe (Nur Werte 0,1 2,3,4)
volatile short p17 = 0;       //Boolsche Variable für Parameter 17 (rlf)
volatile short p8 = 0;        //Boolsche Variable für Parameter 8 (lst)
volatile short p7= 0;         //Boolsche Variable für Parameter 7 (srt)
volatile short p6 = 0;        //Boolsche Variable für Parameter 6 (irt)
volatile int R = 0;        //Zähler für Restlaufzeit Filter
volatile int I = 0;        //Zähler für Istwert Raumtemperatur

// Interrupt-Service Routine fuer RS232-ReceiveDataAvailable
#int_rda
void rdaint_handler () //Interrupt handler. Wenn neue Daten verfügbar sind,
                       //werden diese Daten eingelesen, analysiert, in den String
                       //geschrieben und richtig formatiert in die jeweiligen
```

```
{                       //Variablen geschrieben
   In_Char = fgetc(UZA);// Zeichen einlesen

   if (Auswertung_ok==1) //Haben wir ein gewünschtes Telegramm?
   {
      switch (In_Char)
      {
      case SOT:   RS232_Receive = 1;   //Start of Telegramm -> Wir schreiben in den
                                              String
                  break;
      case EOT:   if ( RS232_Receive == 1)
                  {
                     RS232_Telegram[RS232_Index++]=EOT;//  End of Telegram
                     if (RS232_Telegram[2]==0) //Vor der ID muss 0 im Telegram
                                                     stehen
                     {
                        switch (RS232_Telegram[3]) //An dieser Pos. steht die ID
                        {
                        case 17:
                             T_id = 17; //Telegramm ID = 17
                             break;
                        case 8:
                           T_id = 8;
                          break;
                        case 7:
                           T_id = 7;
                          break;
                        case 6:
                           T_id = 6;
                          break;
                        default:
                           T_id = 0;
                          break;
                         }
                     }

                     RS232_Index = 0;   //String Index auf 0 setzen
                     RS232_Receive = 0;
                     if (T_id == 0)   //kein gewuenschtes Telegramm
                        {
                           Auswertung_ok = 1;
                        }
                        else         //ID erwuenscht, schreibe an WEB
                        {
                           Auswertung_ok = 0;
                        }

                     }
                     break;
                  default:   if ((RS232_Index < 39)&&(RS232_Receive == 1))
                     {
                        RS232_Telegram[RS232_Index] = In_Char ; //Zeichen in String
                                                                  schreiben
                        RS232_Index=RS232_Index+1;             //Index für String
                                                                  erhöhen
                     }
                     break;
      }
   }
}

// Init der Hardware
void HW_Init()
{
// Init der Interrupts
   enable_interrupts(global);
   enable_interrupts(int_rda); // Interrupt, wenn Daten auf der RS232 ankommen
```

```
}

void einlesen()
{
    if (T_id > 0)
   {

      switch (T_id)
      {
      case 17: // Filter Restlaufzeit
            rlf_H = RS232_Telegram[8];   //Restlaufzeit Filter High byte
            rlf_L = RS232_Telegram[9];   //Restlaufzeit Filter Low byte
             rlf = rlf_H;
             rlf = rlf | (((int16)rlf_L)<<8);//Zusammenfügen der beiden Bytes
            R++;      //Zähler (Wie oft haben wir den Parameter schon eingelesen?
            if (R == 5) //nicht jedes Mal ausgeben, sondern nur jedes R-te mal
               {
                R = 0;
                p17 = 1;//Wir schreiben diesen Parameter in der Ausgabe raus
               }
            break;
      case 8:  //Luefterstufe
            lst = RS232_Telegram[8];
            if (lst == 85){ lst = 4;}
            if (lst == 0)
               {
                p8 = 0; //Nullwerte werden nicht raus geschrieben
               }
                else
                  p8 = 1;//Ein wert != 0 wird in der Ausgabe ausgegeben
            break;
      case 7:  //Sollwert Raumtemperatur
            srt = RS232_Telegram[8];
            if (srt == 0)  //Nullwerte werden nicht ausgegeben
               {
                  p7 = 0;
               }
               else
                  p7 = 1;
            break;
      case 6:  //Ist-Wert Raumtemperatur
            irt = RS232_Telegram[8]; //Gleich wie Restlaufzeit Filter, aber mit nur
                                       1 Variablen
            irt = irt | (((int16)RS232_Telegram[9])<<8);
            irt = (irt / 10);
            I++;
            if (I == 3) //nicht jedes Mal ausgeben, sondern nur jedes I-te mal
               {
                I = 0;
                p6 = 1;
               }
            break;
            default:
            break;

      }
      Auswertung_ok = 1;
      T_id = 0;
   }
}

void ausgabe()
{
   if (p17)
     {
       fprintf(WEB, "1%Lu.",rlf);   //Rest- Laufzeit Filter
       p17 = 0;                     // \n = New line \r = Carrige return
     }
```

```
   if (p8)
     {
      fprintf(WEB, "8%u.",lst);   //Lüfter- Stufe
      p8 = 0;
     }
   if (p7)
     {
      fprintf(WEB, "7%u.",srt);   //Sollwert Raum- Temperatur(p6))
      p7 = 0;
     }
   if (p6)
     {
      fprintf(WEB, "6%Ld.",irt);   // Ist-wert Raum- Temperatur
      p6 = 0;
     }
}

// Hauptprogramm
void main()
{
   HW_Init();            //Initialisierung der Hardware

    while (1)
      {
       einlesen();// einlesen der Werte vom Stream UZA
       ausgabe();  //Ausgabe der
      }
}
```

4.4.2 Verwendete Technologien und Entwicklungswerkzeuge

Verwendet wurde das HTL-PIC-Entwicklungsboard mit dem PIC 18f4550, welches durch eine zweite RS232-Schnittstelle erweitert wurde und mit dem PIC Kit2 und der MP-LAB Programmierumgebung programmiert wurde. Hierbei kam der bekannte und bereits in der HTL vorher einmal verwendete CCS-C-Compiler zum Einsatz.

Weiters wurde der SBC65EC-Miniwebserver von Modtronix mit dem PIC18f6627 verwendet. Dieser wurde ebenfalls mit dem MP-Lab programmiert. Da jedoch die mitgelieferte Software des SBC65EC mit dem kostenlosen C18 Compiler geschrieben wurde, habe ich ebenfalls den C18 C-Compiler für die Programmierung des SBC65EC verwendet. Da der SBC65EC mit einem Bootloader geliefert wird, habe ich den Modtronix Network Bootloader verwendet.

Für den Anschluss an ein Gerät muss bei dem SBC65EC ein ausgekreuztes Netzwerkkabel verwendet werden. Man könnte zwar den SBC65EC auch über ein Adapterkabel und den PIC-Kit-2 programmieren, dabei besteht jedoch die Gefahr, den Adressbereich des Bootloaders zu überschreiben.

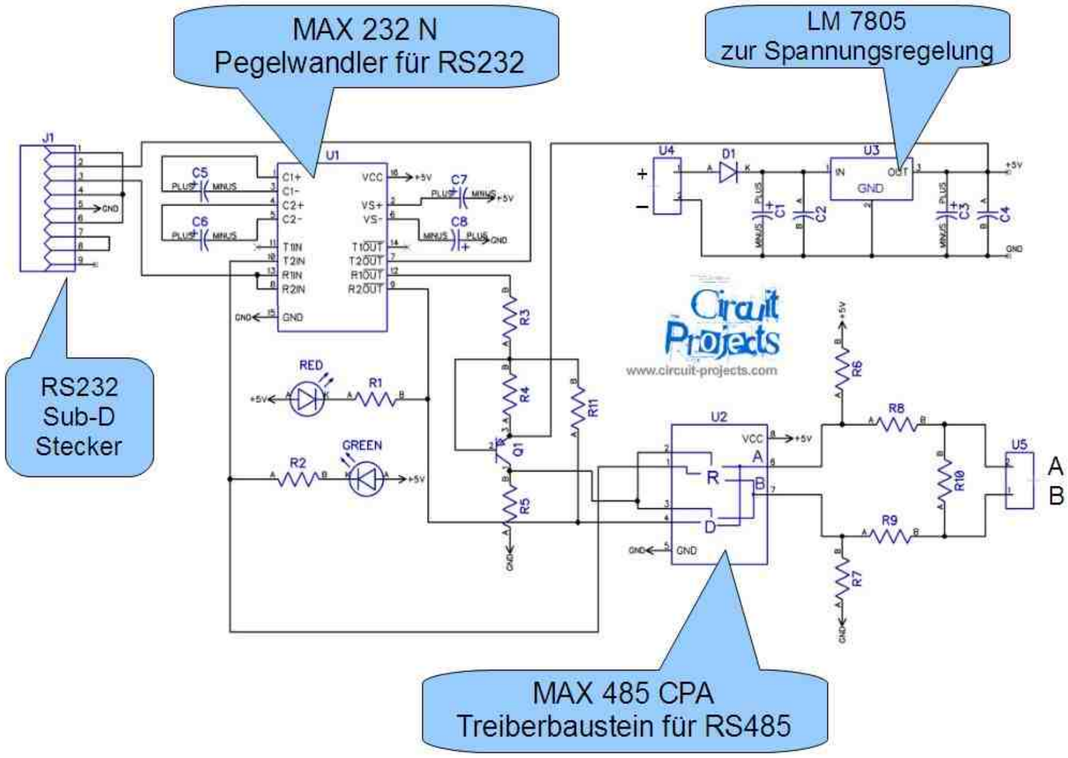
Da die Kommunikation zwischen Bedienteil und der Steuerung des LG500 ein RS485-Bus ist, musste das Signal erst noch auf RS232-Pegel umgewandelt werden, ehe ich es in das

HTL-PIC-Board einlesen konnte. Dabei habe ich lange im Internet recherchiert und Geräte von ca. 20 € bis zu Geräten, die in der Preisklasse um die 500 € liegen, gefunden. Dabei bin ich auf ein Projekt gestoßen, bei dem man sich mit Anleitung einen Pegelkonverter von RS232 auf RS485 und umgekehrt selber bauen kann. Dies war die vielversprechendste und kostengünstigste Variante.

Das Layout der Platine kann man sich ebenfalls herunterladen, ausdrucken, auf eine Leiterplatte aufbügeln und produzieren. Da das Layout schon vorhanden war, war keine „Eagle"-Zeichnung nötig, um die Platine produzieren zu können. Man musste nur beachten, dass der Ausdruck der heruntergeladenen PDF-Datei spiegelverkehrt war und die Schrift auf der fertigen Platine daher spiegelverkehrt sein muss, um die ordnungsgemäße Funktion des Konverters zu gewährleisten.

Zur Datenübertragung werden handelsübliche CAT5-Netzwerkkabel verwendet. Daten werden außerhalb des Gehäuses des Webservers mit RS485 übertragen, um eine möglichst weite, fehlerfreie Datenübertragung zu gewährleisten.

Schaltplan des Konverters von RS485 auf RS232:


MAX 232 N
Pegelwandler für RS232
LM 7805
zur Spannungsregelung
RS232
Sub-D
Stecker
MAX 485 CPA
Treiberbaustein für RS485
Circuit Projects
www.circuit-projects.com
A
B

5 Benutzerdokumentation

Die Websteuerung darf nur von autorisiertem Fachpersonal installiert und in Betrieb genommen werden. Die Konfiguration und Einrichtung des Benutzernamens und Passwortes obliegt ebenfalls nur autorisiertem Fachpersonal (Servicetechniker) und ist im Benutzerhandbuch des LG500 zu dokumentieren.

5.1 Installationsanleitung

Es wird empfohlen nur CAT5-Datenkabel zur Verbindung des Webservers mit dem LG500 zu verwenden. Der Anschlussplan findet sich auf Seite 10 dieser Dokumentation. Der Miniwebserver benötigt je nach Konfiguration 1 oder 2 CAT5-Datenleitungen; 4 Datenleitungen werden parallel zum Bedienteil (A, B, +12 V und GND) angeschlossen. Mit den 8 Leitungen des 2. Datenkabels werden E0, E1, E2 und E3 über die Relaiskontakte der Erweiterungsplatine am SBC65EC verbunden.

Auf der Konfigurationsseite des Miniwebservers (…/xbus.htm) muss eine Datenrate von 9600 eingestellt und bei USART1 das Häkchen gesetzt sein. Anschließend muss der Button „Update“ angeklickt werden, damit die Systemänderungen wirksam werden.

5.2 Referenzhandbuch

Die Menüführung der Webseite ist selbsterklärend. Wenn auf einem Button in der Zeile „Funktionen“ also eine „1“ zu lesen ist und alle anderen Buttons auf „0“ sind, so ist diese Funktion aktiv. Eine Ausnahme bildet die Funktion „Anlage aus/ein“. Die Anlage muss selbstverständlich eingeschaltet sein (Anzeige „0“ auf dem Button G1), um andere Funktionen wirksam schalten zu können.

5.3 Fehlermeldungen und Hinweise auf Fehlerursachen

Sollte der Miniwebserver nicht ordnungsgemäß funktionieren, so überprüfen Sie zuerst, ob die Datenverbindung über Ethernet ordnungsgemäß funktioniert und ob das hausinterne Datennetz (Router etc.) ordnungsgemäß funktioniert.

Im Falle, dass die Daten der Lüftung nicht korrekt auf der Webseite angezeigt werden, zum Beispiel nach einem Stromausfall, so überprüfen Sie bitte die beiden CAT5-Datenleitungen, die den Webserver mit dem LG500 verbinden. Dann schalten Sie auf dem Bedienteil die Funktionen Lüfterstufe und Soll-Temperatur.

Sollte die rote System LED des Webservers nicht regelmäßig blinken, so überprüfen Sie bitte die Spannungsversorgung des Miniwebservers und unterbrechen bitte die Spannungsversorgung des Webservers für ca. 1 Minute.

Falls das Problem nach erneuter Spannungsversorgung des Miniwebservers weiterhin besteht, so kontaktieren Sie bitte einen autorisierten Servicetechniker.

6 Nachprojektphase

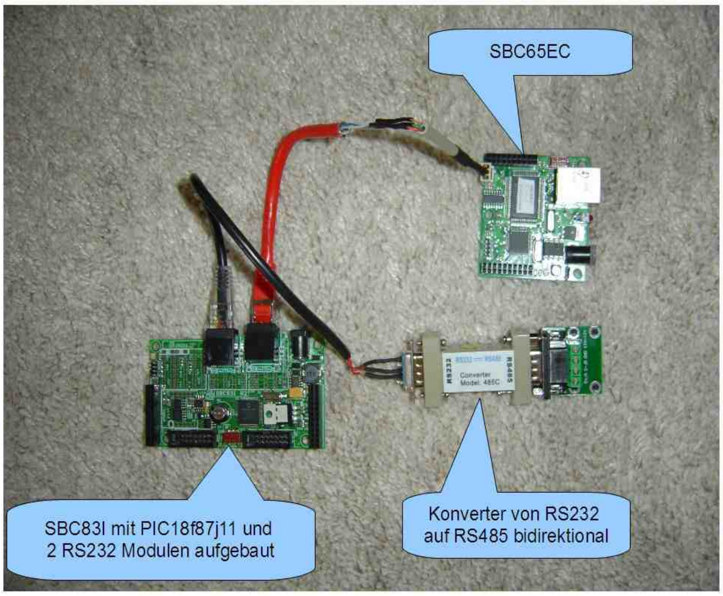
6.1 Nachnutzung

Die Firma Pichler Lufttechnik wird sich entscheiden, ob sie den Miniwebserver in der Praxis einsetzen möchte. Da die Nachfrage nach einem solchen System durchaus gegeben wäre, wäre eine Weiterentwicklung zu einem serienreifen Gerät notwendig.

Da die Lüftungsgeräte der LG-Serie (LG180 und LG250) über ein ähnliches Bedienteil verfügen, sollte es theoretisch möglich sein, den Webserver nur durch Adaptierung der Software auf dem HTL-PIC Board auch für diese Geräte einzusetzen bzw. zu nutzen. Dies müsste jedoch in ausführlichen Tests erst eruiert werden.

6.2 Diskussion

Für eine mögliche zukünftige Serienproduktion habe ich bereits eine Microcontrollerplatine gefunden (SBC83I), die statt dem HTL-PIC Entwicklungsboard zum Einsatz kommen kann, da die gleiche Software in leicht modifizierter Form verwendet werden kann. Die Pinbelegung der RS232-Pins müsste geändert werden. Der Rest könnte gleich bleiben. Sämtliche Teile des Miniwebservers können relativ kostengünstig beschafft werden und einer Serienproduktion steht somit praktisch nichts im Wege.



Im Bild oben ist der Versuchsaufbau mit der möglichen Serienhardware:

Das Gehäuse, an welches keine besonderen Anforderungen gestellt werden, könnte ebenfalls kostengünstig und optisch an den LG500 angepasst in der Firma Pichler hausintern erzeugt werden.

6.3 Arbeitsnachweis Ingenieurprojekt

Robert Stocker				
Datum	**Uhrzeit**	**Beschreibung**	**Während des Unterrichts**	**Außerhalb des Unterrichts**
20.09.2010	18:50–22:00	Recherche Projekt Miniwebserver	03:10:00	
23.09.2010	18:00-22.00	Recherche LG500	04:00:00	
28.09.2010	18:50–22:00	Recherche RS485	03:10:00	
30.09.2010	18:00-22.00	Recherche Datenconverter RS485 RS232	04:00:00	
05.10.2010	18:50–22:00	Recherche SBC65EC	03:10:00	
07.10.2010	18:00-22.00	Recherche RS232	04:00:00	
12.10.2010	18:50–22:00	Vorbereitung Präsentation	03:10:00	
14.10.2010	18:00-22.00	Projektvorstellung	04:00:00	
19.10.2010	18:50–22:00	Recherche Compiler C18	03:10:00	
21.10.2010	18:00-22.00	Recherche Compiler C18	04:00:00	
28.10.2010	18:00-22.00	Recherche SBC65EC	04:00:00	
04.11.2010	18:00-22.00	Recherche SBC65EC	04:00:00	
09.11.2010	18:50–22:00	Recherche RS232	03:10:00	
11.11.2010	18:00-22.00	Recherche Datenconverter RS485 RS232	04:00:00	
16.11.2010	18:50–22:00	Recherche Datenconverter RS485 RS232	03:10:00	
18.11.2010	18:00-22.00	Zusammenlöten des Datenconverters RS232-RS485	04:00:00	
23.11.2010	18:50–22:00	Testen des Datenconverters RS232-RS485	03:10:00	
25.11.2020	18:00-22.00	Auslesen der Daten am Bedienteilbus des LG500	04:00:00	
30.11.2010	18:50–22:00	Analyse der Daten am Bedienteilbus des LG500	03:10:00	
02.12.2010	18:00-22.00	Analyse der Daten am Bedienteilbus des LG500	04:00:00	
07.12.2010	18:50–22:00	Recherche Compiler C18	03:10:00	
09.12.2010	18:00-22.00	Recherche Programmierung des SBC65EC	04:00:00	
14.12.2010	18:50–22:00	Recherche Programmierung des SBC65EC	03:10:00	
16.12.2010	18:00-22.00	Recherche Programmierung des SBC65EC	04:00:00	
21.12.2010	18:50–22:00	Programmierung SBC65EC	03:10:00	
23.12.2010	18:00-21.00	Programmierung SBC65EC	03:00:00	
11.01.2011	18:50–22:00	Programmierung SBC65EC	03:10:00	
13.01.2011	18:00-22.00	Programmierung SBC65EC	04:00:00	
18.01.2011	18:50–22:00	Programmierung SBC65EC	03:10:00	
20.01.2011	18:00-22.00	Analyse Sourcecode von Hermes Electronics	04:00:00	
25.01.2011	18:50–22:00	Analyse Sourcecode von Hermes Electronics	03:10:00	
27.01.2011	18:00-22.00	Vorbereitung der Semesterpräsentation	04:00:00	
03.02.2011	18:00-22.00	Semesterpräsentation	04:00:00	
08.02.2011	18:50–22:00	Recherche Programmierung des HTL-PIC Boards	03:10:00	
10.02.2011	18:00-22.00	HTL-PIC Board Aufbau von USART2 mit MAX232	04:00:00	
22.02.2011	18:50–22:00	Recherche Programmierung des HTL-PIC Boards	03:10:00	
24.02.2011	18:00-22.00	Programmierung des HTL-PIC Boards	04:00:00	
01.03.2011	18:50–22:00	Programmierung des HTL-PIC Boards	03:10:00	
03.03.2011	18:00-22.00	Programmierung des HTL-PIC Boards	04:00:00	
10.03.2011	18:00-22.00	Programmierung des HTL-PIC Boards	04:00:00	
15.03.2011	18:50–22:00	Programmierung des HTL-PIC Boards	03:10:00	
17.03.2011	18:00-22.00	Programmierung des HTL-PIC Boards	04:00:00	
22.03.2011	18:50–22:00	Programmierung des HTL-PIC Boards + Dokumentation	03:10:00	
24.03.2011	18:00-22.00	Test des HTL-PIC Boards + Dokumentation	04:00:00	
29.03.2011	18:50–22:00	Programmierung des SBC65EC	03:10:00	
31.03.2011	18:00-22.00	Programmierung des SBC65EC	04:00:00	
05.04.2011	18:50–22:00	Programmierung des SBC65EC	03:10:00	
07.04.2011	18:00-22.00	Programmierung des SBC65EC	04:00:00	
12.04.2011	18:50–22:00	Programmierung des SBC65EC + Dokumentation	03:10:00	
14.04.2011	18:00-22.00	Aufbau der Relais-Erweiterungsplatine	04:00:00	
28.04.2011	18:00-22.00	Test der Relais-Erweiterungsplatine	04:00:00	
03.05.2011	18:50–22:00	Gesamtsystemtest + Dokumentation	03:10:00	
05.05.2011	18:00-22.00	Einbau in Gehäuse + Dokumentation	04:00:00	
10.05.2011	18:50–19:50	Vorbereitung Projektendpräsentation	03:00:00	
12.05.2010	18:00-22.00	Projekt Endpräsentation	04:00:00	
		SUMME	**198:00:00**	**0**

Gesamtaufwand Robert Stocker: 198 Stunden.

7 Literaturverzeichnis

7.1 Literatur zum Ingenieurprojekt

[LEW 07] „Der Pegelumsetzer MAX232“

Zu finden im Internet unter:

http://www.elektronik-magazin.de/page/der-pegelumsetzer-max232-15

29.10.2007

[PIC 11] Ing. Grassler Wolfgang, BETRIEBS- UND MONTAGEANLEITUNG

KOMPAKTLÜFTUNGSGERÄT LG 500, System VENTECH mit Bedieneinheit „KOMFORT 500-G“, 2010/2011

7.2 Zur Recherche verwendete Webseiten

- http://www.modtronix.com/links/onlinemxws
- http://www.olimex.com/dev/pic-web.html
- http://www.modtronix.com/products/sbc65ec/sbc65ecr201.pdf
- http://www.ribu.at/OnLi_Kat.htm
- http://4--all.com/howtos/how-to-code-a-submit-button-on-a-web-page.html
- http://www.digitale-elektronik.de/shopsystem/
- http://www.modtronix.com/picboards/prog/
- http://www.circuit-projects.com/converter-circuits/rs232-rs485-converter-with-automatic-rx-tx-control.html
- http://www.axotec.de/produkte/embedded-computer/rs485rs422-ethernet.html
- http://www.zen74158.zen.co.uk/kk-files/k2.pdf
- http://www.modtronix.com/soft/netloader/
- http://modtronix.com/products/sbc65ec/websrvr65_v310/mainpage.php?mainpagehtml=page_webpages&mainpageName= (Webserver)
- http://www.modtronix.com/soft/mxd/help/index3.htm (Debugger)
- http://modtronix.dyndns.org/index.htm (Demo-Webserver)
- http://de.wikipedia.org/wiki/Integer_(Datentyp)#Darstellungen_von_Integern
- http://de.wikipedia.org/wiki/zweierkomplement

8 Funktionsbeschreibung LG 500

Die Funktionsbeschreibung wurde der Bedienungsanleitung des LG 500 der Firma Pichler Lufttechnik entnommen.

[...“Das Kompaktlüftungsgerät LG 500 System VENTECH ist zum Einbau in raumlufttechnischen Anlagen für die kontrollierte mechanische Be- und Entlüftung von Wohnungen und von Räumen mit ähnlicher Zweckbestimmung wie z.B. Seminarräume und Kleinbüros mit einem maximalen Luftvolumenstrom von 550 m³/h geeignet.

Durch den Einbau einer mechanischen, kontrollierten Lüftung für Wohnungen wird der gesamte Wohnbereich mechanisch be- und entlüftet. Dabei ist in den Zuluftbereichen die kontrollierte Luftversorgung mit aufbereiteter und gefilterter Außenluft gegeben. Im Abluftbereich werden Gerüche und die feuchte Raumabluft abgeführt.

Zweck der kontrollierten mechanischen Be- und Entlüftung von Wohnungen ist die Verbesserung der Luftqualität, die Verringerung des Heizenergiebedarfs durch den Einsatz eines hocheffizienten Wärmerückgewinnungssystems sowie die Beeinflussung der Raumluftfeuchte.

Der Anwendungsbereich und die bestimmungsgemäße Verwendung für das Gerät beschränkt sich auf den Einsatz in raumlufttechnischen Anlagen zur Absaugung von verbrauchter Luft und zur Zuführung frischer, temperierter Außenluft bei maximalen Fördermitteltemperaturen der Luft von -15 °C bis +35 °C. Des Weiteren muss die geförderte Luft frei von aggressiven Dämpfen und von verschleißfördernden Stoffen sein.Jede andere Anwendung gilt als zweckentfremdet....

...“Bei der mechanischen kontrollierten Wohnungslüftung wird die verbrauchte, feuchte Abluft aus den Nassräumen der Wohnung, wie z.B. Bad, WC und Küche abgeführt und gegen aufbereitete frische und gefilterte Außenluft in den Aufenthaltsbereichen wie z.B. im Wohn-, Schlaf- und Aufenthaltszimmer ausgetauscht.

Durch einen bedarfsgeführten Anlagenbetrieb, den Einsatz eines hocheffizienten Wärmeaustauschers zur Rückgewinnung der Wärme aus der Abluft in die Zuluft und unter Verwendung stromeffizienter Ventilatoren mit neuester EC-Technologie lässt sich eine hohe Energieeinsparung im Betrieb der Lüftungsanlage realisieren.

Dabei gilt es besonders zu beachten, dass der hocheffiziente Wärmeaustauscher mittels einer geeigneten geregelten Frostschutzstrategie gegen Einfrieren geschützt wird und u.a. ein wirksamer Kondensatwasserablauf sicherzustellen ist.

Je luftdichter die Gebäudehüllen ausgeführt und je wirksamer ein Wohnhaus gedämmt wird, umso mehr lohnt sich diese Technik....“

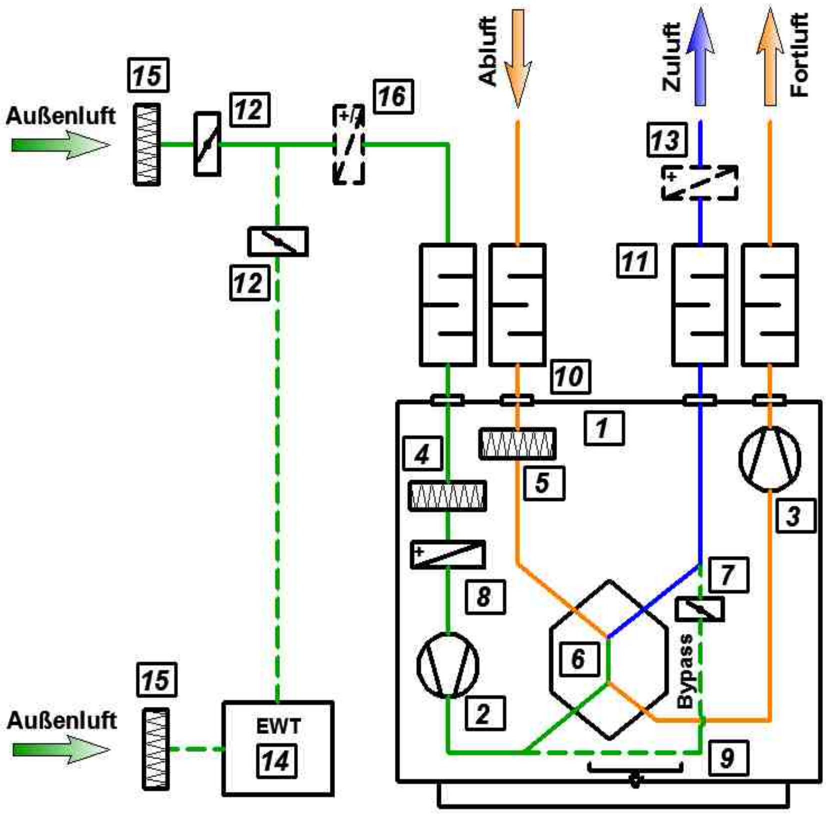


Bild 1 - Prinzipschema über Systembauteile

Im Prinzipschema ist der grundsätzliche Aufbau des Wohnungslüftungsgerätes mit den optionalen zusätzlichen Systemerweiterungen dargestellt.

1 - Gerätegehäuse

2 - Zuluftventilator in EC-Ausführung

3 - Abluftventilator in EC-Ausführung

4 - Außenluftfilter für Feinstaub F7 (optional als Pollenfilter Güteklasse F9)

5 - Abluftfilter G4 für Grobstaub

6 - Gegenstromwärmetauscher

7 - Bypassklappe

8 - Frostvorheizung für Gegenstrom - Wärmetauscher (optional)

9 - Kondensatwasserwanne mit Abfluss und bauseitigem Geruchsverschluss (Siphon)

10 - Anschlussstutzen am Lüftungsgerät

11 - Schalldämpfer (optional) im Leitungssystem

12 - Umschaltklappe (optional)

13 - Nachheizregister Ausführung mit Warmwasser (optional) mit nachgeschaltenem Zulufttemperaturfühler

14 - Erdwärmetauscher (optional)

15 - Ansaugelement mit Vorfilter G4

16 - Heiz-, Kühlregister für Sole-EWT (optional)

…]

[PIC, 11]